

ASCII
ASCII CORPORATION

MSX-DOS™ TOOLS

エムエスエックスドスツールズ



エムエスエックスドスツールズ
MSX-DOS™ TOOLS

ASCII
ASCII CORPORATION

MSX-DOSTM TOOLS

USER'S MANUAL

ASCII
ASCII CORPORATION

- ・MSX, MSX-DOS, MSX・M-80, MSX・L-80はアスキーの商標です。
- ・MS-DOS, XENIXは米国マイクロソフト社の商標です。
- ・Z80は米国Zilog, Inc. の商標です。
- ・UNIXオペレーティングシステムは米国AT&Tのベル研究所が開発し, AT&Tがライセンスしています。
- ・CP/Mは米国デジタルリサーチ社の商標です。

はじめに

MSX-DOS TOOLSは、高度な機能を持ったMSX-DOSに加え、28のツールソフトウェア、強力なスクリーンエディタ、および大型コンピュータ上のソフトウェアに匹敵する機能を持ったユーティリティソフトウェアをパッケージしたプロ必携のソフトウェアです。MSX-DOSがさらに使いやすくなるうえに、生産性を高め、アセンブリ言語によるアプリケーション・プログラムの開発環境を大幅に向上させるソフトウェアです。

このユーザーズマニュアルには以下のものが含まれています。

- ・ MSX-DOS TOOLS マニュアル
- ・ MSX-DOS マニュアル
- ・ MED (MSX-DOS スクリーンエディタ) マニュアル
- ・ MSX-DOS ユーティリティソフトウェア マニュアル

ご注意

- (1) このソフトウェアならびにマニュアルを賃貸業に使用することを禁じます。また、このソフトウェアならびにマニュアルの一部または全部を無断でコピーすることはできません。
- (2) このソフトウェアは、製品登録カードを返送して当社に登録済の方に限り、使用したりコピーしたりすることができます。なお、このソフトウェアを個人使用以外の目的でコピーすることはできません。
- (3) このマニュアルに記載されている事柄は、将来予告なく変更することがありますが、当社に登録されている方には案内をお送りします。
- (4) 製品の内容については万全を期しておりますが、製品の内容についての御不審や、誤り、マニュアルの記載もれなど、お気づきのことがございましたら、マニュアル巻末の「お問い合わせ」についての要領で下記の問い合わせ先へお送り下さい。
- (5) このソフトウェアを運用した結果の影響については、(4)項にかかわらず、責任を負いかねますので御了承下さい。

お問い合わせ先 ☎ 107 東京都港区南青山6-11-1 スリーエフ南青山ビル

株式会社 アスキー ユーザーサポート係

TEL 03-498-0205 (祝祭日を除く月～金)
10:00～12:00, 13:00～17:00

MSX-DOS-TOOLS

目 次

1	MSX-DOS TOOLSの概要	P 1
1.1	UNIXの入出力環境とMSX-DOS	P 1
1.2	MSX-DOS TOOLSの標準入出力	P 1
2	入力と出力	P 2
2.1	I/Oリダイレクション	P 2
2.2	パイプ機能	P 2
2.3	シーケンシャルプロセス	P 3
3	コマンドの書式指定	P 3
3.1	ファイルスペック	P 3
3.2	ファイル名	P 3
3.3	ドライブ名	P 3
3.4	ファイルの間接指定	P 4
3.5	西暦指定	P 5
3.6	パラメータの区切	P 5
4	ツールコマンド一覧	P 6
5	コマンドリファレンス	P 7~40
5.1	コマンドリファレンス表示オプション“/H”について	P 7
6	エラーメッセージ一覧	P 41
付録	TOOLS使用サンプル集	P 45

1 MSX-DOS TOOLSの概要

MSX-DOS TOOLSはMSX-DOS上でのプログラム開発を強力にサポートする外部コマンド群です。TOOLSを構成するソフトウェアは、28のCOM形式のファイルからなり、DOS上でコマンド行を入力することによりDOSの内部コマンドと同様の操作感覚で実行できます。

個々のコマンドはディスク管理、ファイル操作からカレンダーや電卓機能等、幅広くサポートし、また、パイプライン、I/Oリダイレクション機能（後述）によりUNIXライクな入出力操作が可能です。

1.1 UNIXの入出力環境とMSX-DOS

UNIXオペレーティングシステムがプログラムの開発環境として、その実力を高く評価される理由の一つとして、豊富なツール（UNIXではソフトウェアを何らかの目的を達成するための道具としてとらえる立場から個々の開発支援、その他のプログラムをこう呼ぶ）を機能や目的に応じて縦横に組合せることができるという点があります。UNIXの環境下では、あるプロセス（個々のプログラムの実行過程）と他のプロセスの間でデータの受け渡しを行ったり、複数のプロセスをそれぞれ独立にディスクファイルやコンソールから入力を行ったり、プリンターやディスクファイルにデータを出力したりしながら、順次実行していくなどといった柔軟性に富んだ入出力操作が可能です。

こうしたすぐれた入出力環境を実現する共通のインターフェースを、UNIXではOSレベルでサポートし標準入出力として各プログラムが用いることができますが、残念ながらMSX-DOSには標準入出力の概念はなく、UNIXのような入出力環境をOSのレベルではサポートしていません。

1.2 MSX-DOS TOOLSの標準入出力

MSX-DOS TOOLSでは、UNIXライクな操作環境を実現するため、UNIXの標準入出力をコマンド内部でシミュレートしています。

MSX-DOS TOOLSで用いられる標準入出力は以下のような機能をサポートしています。（各機能の詳説は、2.入力と出力をご覧ください）

a) コンソール入出力

通常、入力はキーボードから、出力はスクリーンに表示します。

b) I/Oリダイレクション

入出力をディスクファイルや他のデバイスに切り換えます。

c) パイプ機能

あるコマンドの出力から別のコマンドの入力へデータの受け渡しを行います。

d) シーケンシャルプロセス

複数のコマンドをコマンド間のデータの受け渡しなしにそれぞれ独立に実行します。

2 入力と出力

多くのツールコマンドでは前述の機能を持った標準入出力がサポートされています。コマンドへの入力をキーボードから行ったり、あるいはスクリーンに結果を表示するといった通常の入出力の他に後述のリダイレクトを用いて入力をキーボードからディスク上のファイルに切り替えたり、出力をスクリーンでなく、ディスクファイルやプリンタへ転送することができます。また、パイプ機能（後述）により、あるコマンドの出力をそのまま別のコマンドの入力にあてるといったUNIXライクな入出力環境を実現することが可能です。

2.1 I/Oリダイレクション

リダイレクションにより標準入出力を通常のコンソール入出力からプリンタやファイル等に転送することができます。リダイレクトには“<”、“>”の記号を用います。

例1

```
CAL >PRN
```

では、カレンダーをプリンターへ出力し、

例2

```
SORT <LIST1 >B:LIST2
```

ではカレントドライブのLIST1というファイルをソートし、BドライブにLIST2という名でファイル出力します。

さらにPATCHコマンドでは以下のようにバイナリファイルの更新データをファイルから受け取るによりキー入力なしに確実なファイル更新が可能です。

```
PATCH FILE.BIN <EDIT.DAT
```

この例ではFILE.BIN というファイルをディスク上のEDIT.DATというファイル中の更新データにしたがって更新します。（詳細はPATCHコマンド参照のこと。）

2.2 パイプ機能

あるコマンドプロセスの標準出力を次のプロセスの標準入力に送りこむという操作をサポートします。

例

```
SORT <LIST1 | HEAD
```

ではカレントドライブのLIST1というファイルがソートされその出力がHEADコマンドの入力となり先頭部分が画面出力されます。

パイプ機能ではディスク上に中間ファイルがつくられるためディスクがライトプロテクトされた状態で使用しますと処理を中断します。通常はカレントドライブに中間ファイルはつくられますが、“|a ”のようにブランクをあけずにドライブ名を指定することにより、つくられるドライブを変更することができます。

また、プロセス間の入出力の受け渡しをせずに個々のプロセスを独立に順次実行する場合には次に述べるシーケンシャルプロセスを用います。

2.3 シーケンシャルプロセス

複数のコマンドプロセスを順次実行します。パイプラインのように前のプロセスの標準出力が後のプロセスの標準入力となることはありません。

各プロセスは“;”でつなぎます。

例

```
HEAD LIST ; SORT LIST
```

では、HEADコマンドがLISTというファイルの先頭部分を画面表示したのち同じファイルLISTをあらためてディスクから読みだしソートして画面に出力します。

注) 以上述べた入出力環境は、TOOLSのコマンドレベルでサポートされているものであり、DIR、TYPE等のMSX-DOSの内部コマンドでは使用できません。

3 コマンドの書式指定

ツールコマンドの書式表記法はDOSの内部コマンドの表記に準拠しますが、以下いくつか補足します。(DOSマニュアル第2部参照)

3.1 ファイルスペック

書式中で“ファイルスペック”となっているパラメータはドライブ名、ファイル名、ピリオド、拡張子名が正しい順序で並んだものを示し、ファイル名、拡張子名はワイルドカードキャラクタ(*, ?)を含むことができます。

さらに上記の指定を“+”または“, ”でつなぐことにより複数指定可能です。(間のブランクは無視されます)

ドライブ名は常に省略可能であり、デフォルトはカレントドライブとなります。また、ファイルスペックのかわりにファイルの間接指定(3.4 ファイルの間接指定を参照のこと)を用いることが可能です。

例

```
b:*.bat+*.c+a:studio.h
```

3.2 ファイル名

“ファイル名”で示されるパラメータはワイルドカードキャラクタを含むことができないという点、複数の指定がゆるされないという点でファイルスペックと区別されます。その他のについては同様の指定となります。

3.3 ドライブ名

“ドライブ名”で示されるパラメータは接続されているドライブのA～Hまでの各ドライブ名にコロン“:”をつけたもので示します。

3.4 ファイルの間接指定

ファイルの間接指定とは、複数のファイルを同時に指定可能なコマンドについて、前述の“ファイルスペック”を用いず必要なファイル名をまとめてひとつのファイルにして、コマンド行ではその“ファイル名のファイル”を間接的に指定することです。

ファイルの間接指定により複数のファイル指定を簡単なコマンド行の入力でおこなうことができます。

(1) 間接指定の方法

通常のファイルスペックを指定するかわりに下記のフォーマットにしたがったファイル名のリストファイルを“[]”で囲んで指定します。コマンド側は“[]”で囲まれたファイルを間接指定用のリストファイルとしてオープンしその内容を通常のファイル指定と同様に処理します。

間接指定の“[]”のなかでワイルドカードを用いたり複数のリストファイルを指定することはできません。また、間接指定をもちいた場合はそのみが有効であり、同時にファイルスペックやファイル名の指定をしても無効となります。

(2) ファイル名のリストファイルについて

ファイルの間接指定のためのファイル名のリストファイルは以下のフォーマットで作ります。

- ・必要なファイル名を1行にひとつだけ指定します。指定できるファイル数に制限はありません。
- ・ファイル名にワイルドカードはもちいられません。
- ・各行頭のスペース、タブは無視され最初のキャラクターからファイル名とみなされます。

(3) 間接指定の用例

間接指定の用例としてディスク上の拡張子“bat”のついたすべてのファイルをファイルサイズの更新日の新しい順にそのヘッダー部分を見ながら必要なら別のディスクにコピーするという操作を以下に解説します。

- 1) LSコマンドのリダイレクトによりリストファイルをつくる。

```
ls /t/c *.bat >b:bat.lst
```

LSコマンドの/T、/C オプションによりカレントドライブのディスク上の拡張子“bat”をもったすべてのファイルが最新更新日の新しい順にソート（/Tオプションによる）され一行に一ファイル名のリストが（/Cによる）Bドライブにbat.lstという名でリダイレクトされます。

- 2) bat.lstをファイル間接指定のリストファイルに用いてMENUコマンドにより必要なファイルをコピーする。

```
menu /t [b:bat.lst]
```

これにより拡張子“bat”をもったファイルが更新日の新しい順にプロンプト表示され以下MENUコマンドのTコマンドによりそのヘッダー部分を見なが

ら、必要ならCコマンドでコピーすることができます。

3.5 西暦指定

西暦年 (yyyy) の入力は、4桁のうち上2桁が省略可能でありデフォルトは20世紀となります。

00~99 ...1900~1999年

3.6 オプションスイッチ

“/” にアルファベット1文字をつけた (“/S100” のようにパラメータを伴うこともある) アーギュメントは、コマンドの各種のオプション機能を実行するためのスイッチを意味します。アルファベットは大文字でも小文字でも有効であり、また複数のオプションスイッチを指定する場合、スペース、タブ等で区切らずに続けて指定可能です。

例

① GREP/X/N/A----

② g r e p / x _ / n _ / a ----

①、②どちらも有効です。

3.7 アーギュメントの区切り

ツールコマンドでは、書式中 “_” であらわされるアーギュメントの区切りにはスペースまたはタブのみが有効です。カンマを用いることはできません。

但し “/X” のようなオプションスイッチを複数指定する場合各オプション間のスペースは省略できます。

例

g r e p / x / n / a _ “----” _ < x x x . c

4 ツールコマンド一覧

BEEP	BEEP音の発生
BIO	バイオリズムの出力
BODY	ファイルの一部の切出し
BSAVE	HEXファイルのバイナリファイルへの変換
CAL	カレンダーの表示
CALC	簡易電卓
CHKDSK	ディスクの状態の表示
CLS	クリアスクリーン
DISKCOPY	バックアップコピー及び照合
DUMP	ファイルの16進ダンプ
ECHO	コマンド行の表示
EXPAND	スペース・タブの変換
GREP	任意の文字列の検索
HEAD	ファイルの先頭部分の表示
HELP	コマンドリファレンスの表示
KEY	ファンクションキーの設定
LIST	BASICの中間言語ファイルのソースファイルへの変換
LS	ディレクトリの詳細情報の表示
MENU	ディレクトリのファイルの諸操作
MORE	ファイルの表示
PATCH	バイナリファイルの更新
SLEEP	アラーム機能
SORT	ソート (クイックソート)
TAIL	ファイルの末尾部分の表示
TR	文字列の置き換え
UNIQ	重複行の削除
VIEW	ファイルの表示 (スクロール)
WC	語数、行数、ページ数カウント

5 コマンドリファレンス

5.1 コマンドリファレンス表示オプション “/H” について

すべてのツールコマンドで “/H” オプションを用いることによりコマンドリファレンスを表示、参照することができます。但しこの場合コマンドは実行されません。

以下ツールコマンドをアルファベット順に解説します。

BEEP

書式

BEEP [/H]

機能

BEEP音を発生します。

解説

BEEP音を発生します。

バッチ処理中などで用いることにより、処理の進行の目安とすることができます。

また、BEEPコマンドでは、リダイレクトによりファイルにBELコード（16進の07 h）を送ることも可能です。

例

```
A>beep␣  
Beeeeeeeeep !  
A>
```

B I O

書式

B I O [/ H] [< y y y y / m m / d d >] [{ [[y y y y /] m m /] d d }]

機能

バイオリズムを出力します。

解説

出力は標準出力です（通常はC R Tディスプレイ）

生年月日を指定すると、当日と前後20日のバイオリズムが表示されます。生年月日の指定は 西暦年, 月, 日の順にスラッシュで区切り指定します。

オプションとして、年月日を指定すると、指定の日と前後20日間のバイオリズムが出力されます。この場合、年, 月のデフォルトは当年, 当月で省略可能ですが、月のみの省略は許されません。

例

A>bio 1965/3/30 ↵

BODY

書式

```
BODY [/H] [/N] [/S 〈開始行〉] [/E 〈終了行〉] [
    [ 〈ファイルスペック〉 ]
```

機能

入力の一部を切り取って出力します。

解説

入出力は通常標準入出力が用いられますが、入力については〈ファイルスペック〉による、1つまたは複数のファイルの指定が可能です。

出力には入力の先頭行からの行番号、ファイル名がつきます。

出力の開始行、終了行は以下のスイッチにより指定します。

“/S” 入力の切り出し開始を〈開始行〉で指定します。

“/E” 切り出し終了行を〈終了行〉で指定します。

以上2つのデフォルトはそれぞれ、入力の先頭、末尾行となります。

“/N”を指定すると、行番号、ファイル名等の出力が省かれ、行本体のみの出力となります。

例

```
A>body test.c /s100 /e120 』
    TEST.C:
    100: VOID fnk1(arg1,arg2,arg3)
    101:         int arg1,arg2,arg3;
           |
    120: } /*end of fnk1 */
A>
```

BSAVE

書式

BSAVE [/H] [<ファイル名1> [<ファイル名2>]

機能

HEX形式のファイルをバイナリファイルに変換します。

解説

アセンブラ、リンカー等により生成したHEX形式のファイルを入力することにより、BASICの“BLOAD”命令でロード、実行が可能なバイナリファイルを作成します。これにより、従来の煩雑なファイル操作なしに、マシン語のルーチンがBASICの環境下で実行可能となります。

入力には必ずディスク上のHEXファイルを指定し、出力は標準出力または<ファイル名2>の指定によりディスクファイルへの出力も可能です。

入力にHEX形式以外のファイルを指定すると、“<ファイル名> is not a hex file”のエラーとなります。

例

```
A>BSAVE TEST.HEX >TEST.BIN ↵  
complete  
A>
```

書式

CAL [/H] _ [/Y < y y y y > | < [y y y y /] mm >]

機能

カレンダーを出力します。

解説

出力は標準出力が用いられ、通常画面表示となります。

以下オプションの指定とその機能について解説します。

- ① 月を指定することにより、指定の月とその前後1ヶ月のカレンダーを出力します。
月の指定は < y y y y / mm > で年、月をスラッシュで区切って行います。
各々のデフォルトは当年当月で省略可能ですが、月のみの省略は出来ません。
- ② “ /Y ” スイッチで年を指定した場合、 < y y y y > 年の1年分のカレンダーを出力します。
年の指定は上2桁を省略出来ます。

例

A>CAL 2 ♪

Jan 1987							Feb 1987							Mar 1987						
Su	M	Tu	W	Th	F	Sa	Su	M	Tu	W	Th	F	Sa	Su	M	Tu	W	Th	F	Sa
					1	2	1	2	3	4	5	6	1	2	3	4	5	6	7	
4	5	6	7	8	9	10	8	9	10	11	12	13	8	9	10	11	12	13	14	
⋮																				

CALC

書式

CALC [/H] _ [〈演算式〉]

機能

簡易電卓機能を提供します。

解説

四則演算を通常の演算書式による計算式の入力によりおこなうことができます。

CALCコマンドでは演算式の入力に二つの方法をサポートしています。

1) コマンド行で式を指定しない場合 (連続計算可)

“CALC”と入力しコマンドを起動したのち、計算したい数式を入力しリターンキーをおすと改行し計算結果を表示するとともに次の計算式の入力待ちの状態となり以下CTRL-C、CTRL-Z、“Q”または“q”によりコマンドをぬけるまで連続して計算をおこなうことができます。

2) コマンド行で式を指定する場合

コマンド行で計算式を入力することも可能です。但しこの場合1回の計算でコマンドを終了します。

CALCコマンドでは、十進十二桁の精度での浮動小数点演算をサポートし、 $-9.999999999999E+99 \sim 9.999999999999E+99$ の範囲の数値を扱うことができます。

例

```
A>CALC 12+1222.2+345/2↵
```

```
1406.7
```

```
A>
```


CHKDSK

書式

CHKDSK [/H] [/F] [/M] _ [<ドライブ名>]

機能

ディスク上のファイルの状況、ディスクの残り容量および、オプションによりメディアに関する情報を表示します。

解説

<ドライブ名> で指定したドライブ（省略はデフォルトドライブ）のディスクをチェックし、以下の内容について報告します。

- ・ 全ディスク容量
- ・ ディスク上のファイル数とトータルファイルサイズ
- ・ 使用可能ディスク容量
- ・ ディスク上のファイル状況に誤りがあると、それを表示します。

つぎに示すエラーは、“/F” スイッチを指定した場合、自動的に修復されます。

```
<ドライブ名> has invalid cluster, file truncated
```

つぎのエラーは、ユーザーが処理を選択してください。

```
<XXXX> lost clusters found in <YYY> chains
```

“/F” スイッチが指定されていると、

```
Convert lost chains to files (Y/N)
```

と表示されます。

この表示に対して、“Y” を選択した場合には

ファイル名 F I L E n n n n . C H K を作成し、次のような表示をします。

```
<n n n n> bytes disk space freed
```

また、“N”を選択した場合には、

```
<nnnn> bytes disk space would be freed
```

と表示されます。

```
<ファイル名> is cross linked on cluster <nnnn>
```

と表示されたら、それぞれ必要なファイルのコピーを作成した後、重複してリンクしている元のファイルを消去してください。

さらに“/M”スイッチにより

- ・ クラスタサイズ及び数
- ・ セクタサイズ及び数
- ・ ディレクトリエントリーの数

などのメディアに関する詳細を表示します。

例

```
A>chkdsk b: /m
512 bytes per sector
1* first FAT sector
3 sectors per FAT
2 copies of FAT
112* first directory sector
112 directory entries
1024* first data sector
1024 bytes per cluster
713 total clusters
713K total disk space
683K in user files
75K available disk space
```

A>

CLS

書式

CLS [/H]

機能

スクリーン画面をクリアします。

解説

CLSコマンドでは標準出力にCtrl-L (0CH) コードを送ることにより画面をクリアします。そのためリダイレクトによりファイルに0CHコードを送ることも可能です。

DISKCOPY

書式

DISKCOPY [/H] [/V] [/C] _ [〈ドライブ名1〉] _
[〈ドライブ名2〉]

機能

ディスク単位でのバックアップコピー及び照合を行います。

解説

〈ドライブ名1〉で指定されたドライブのディスクの内容をすべて〈ドライブ名2〉のディスクにコピーします。

DISKCOPYコマンドにより出来た2枚のディスクは同一のファイル内容を持ちます。COPYコマンドと違い、不連続のファイルを連続したファイルにコピーすることは出来ません。また、〈ドライブ名2〉のディスクの元のファイルは、DISKCOPYによってすべて失われてしまいます。

属性の異なるメディア間でのコピーは出来ません。

ディスクドライブ名のデフォルトは1. 2ともそれぞれカレントドライブとなります。

- ・ “/V” スイッチを指定するとディスクのコピーではなく照合のみを行います。
- ・ “/C” スイッチを指定するとコピーの実行に続いてディスクの照合を行います。

例

```
A>diskcopy a: b: ↵
Source disk a:
destination disk b:
strike a key when ready ↵
all data and file on destination disk
will be erased!
O K ? ↵
complete!
copy other disk (Y/N) ? N ↵
A>
```


DUMP

書式

```
DUMP [/H] [/N] _ [/B <1行のバイト数>] _ [/S <開始バイト>]  
_ [/E <終了バイト>] _ [ <ファイルスペック> ]
```

機能

ファイルを16進数及びキャラクタによりダンプします。

解説

入出力には通常標準入出力が用いられますが、入力には〈ファイルスペック〉による1つまたは複数のファイルが指定可能です。

ダンプには、16進数とASCIIキャラクターが用いられ、16バイト毎に改行し、標準出力に表示されます。さらに、“/B”スイッチにより行毎のバイト数をユーザーが指定可能です。

“/S” “/E” の各スイッチにより、ダンプの開始バイト、終了バイトをそれぞれ16進数で指定出来ます。デフォルトはファイルの先頭及び末尾です。

“/N” はファイルのアドレス番地及びキャラクターダンプを省き16進数ダンプリストのみを出力します。“/B” オプションとの組合せで1行のバイト数を変更することにより、例えば1行に1バイトずつ出力しリダイレクトによりファイルにするなどの操作が可能です。

例

```
A>DUMP ABC.COM /EF8 𐀀  
0000 : C3 26 12 11 F9 FF CD 8A 26 21 13 00 39 5E 23 56      .&..y...&!..9^#V  
0010 : 21 01 00 CD F5 27 CA 02 02 21 13 00 39 5E 23 56      !...u'...!..9^#V  
0020 : 21 02 00 CD 21 28 CA 7C 01 21 01 00 EB 21 08 00      !...!(. |.!...!..  
.  
.  
.  
00F0 : 39 E5 CD 25 0C D1 D1 22 39                          9..%..."9
```

ECHO

書式

ECHO [/H] [〈アーギュメント〉]

機能

コマンドアーギュメントを標準出力に出力します。

解説

コマンド行で指定したアーギュメントをそのまま標準出力に出力します。

例

```
A>echo hello
hello
A>
```

EXPAND

書式

EXPAND [/H] [/R] [/F] _ [<ファイルスペック>]

機能

スペース・タブの変換をします。

解説

オプションを省略した時、タブをスペースに変換します。

“/R” オプションにより、反対にスペースをタブにまとめることが出来ます。さらに、“/F” オプションをもちいることにより各行頭から最初のキャラクター（スペース、タブ以外の）間のみで変換をおこなうことが可能です。これにより例えば、プログラム中の文字列データ内のスペース、タブが変換されてしまいコンパイル後、実行時に予想外の表示がなされるといったトラブルを防ぐことができます。

書式

```
GREP [/H] [/X] [/C] [/N] [/A] _ "〈文字列〉" _  
[ 〈ファイルスペック〉 ]
```

機能

入力から任意の文字列を検索し、出力します。

解説

入出力は通常標準入出力が用いられますが、入力には〈ファイルスペック〉による1つまたは複数のファイルが指定可能です。

文字列はダブルクォーテーションで囲み必ずファイルスペックの前で指定します。

入力中の検索文字列を含む行を先頭からの行番号をつけて出力します。入力がファイルスペックの場合、ファイル単位でファイル名とともに出力されます。

文字列中で大文字、小文字は区別されません。

GREPコマンドには多くのスイッチがあり、これらを有効に組み合わせることにより、コマンドを効果的に用いることが出来ます。

- ・ “/X” 指定した文字列を含まない行を返します。
- ・ “/C” 指定した文字列を含む行数のみを返します。
- ・ “/N” 行番号、ファイル名を省きます。
- ・ “/A” 大文字、小文字を区別して検索します。

例

```
A>grep "fprintf(" FILE.c  
A:FILE  
7:      fprintf(stderr,"%s",MESS1);  
8:      fprintf(stderr,"%d line contain ...  
      .....  
21:      fprintf(stderr,"Unknown switch...  
9 Lines contain "fprintf("  
A>
```


HEAD

書式

HEAD [/H] [/N] [/L 〈出力行数〉] _ [〈ファイルスベック〉]

機能

入力先頭から指定された行を切り取って出力します。

解説

入出力は通常標準入出力が用いられますが、入力については〈ファイルスベック〉で1つまたは複数のファイルが指定可能です。

入力先頭行から指定の行数を切り取り、行番号、残り行数、入力がファイルスベックである場合、ファイル名をつけて出力します。この際各行は、コンソール画面幅を超えた部分については出力されません。

出力の内容は以下のスイッチにより変更可能です。

- ・ “/L” 先頭から何行出力するかを〈出力行数〉で指定します。
指定のない場合、先頭から10行分の出力となります。
- ・ “/N” スイッチを指定すると、行番号、ファイル名等すべて省かれ、行本体のみの出力となります。この場合の出力は画面幅と無関係に全て表示されます。

例

```
A>head test.doc ↵
```

```
TEST.DOC :
```

```
1: 1st line
```

```
2: 2nd line
```

```
⋮
```

```
10: 10th line
```

```
A>
```

HELP

書式

HELP <コマンド名>

または、

<コマンド名> /H

機能

ツールコマンドのリファレンスを出力します。

解説

MSX-DOSツールのコマンドすべてについてHELPによって参照することができます。

(但し、HELPコマンドでMSX-DOSの内部コマンドを参照することはできません)

また、各コマンドで“/H” オプションを指定することのによっても同様のヘルプメッセージを表示します。

例

```
A>help key
```

```
MSX-DOS TOOLS HELP KEY
```

```
Displays or erases function ..
```

```
'ON/OFF' ..
```

```
...
```

```
Usage: key /h .....
```

```
...
```

KEY

書式

KEY [/H] [/I] [ON|OFF]
[〈ファンクションキー番号〉, 〈設定文字列〉] ……

機能

ファンクションキーに文字列を設定します。

解説

画面最下行に現在のファンクションキーの設定を表示 (KEY ONの状態) したり、消去 (KEY OFF) したりすることができます。

コマンド行で変更したいファンクションキー番号 (1~10) に続いて、設定文字列を入力することにより、ファンクションキーの設定をします。キー番号と文字列はコンマで区切ります。

設定文字列中に “CR” “ESC” などのコントロールコードを入力したい場合、“¥hh” のように “¥” に続けて2桁の16進数でキャラクターコードを入力します。

設定文字列の長さは15文字まで指定可能ですが16文字以上の文字列を指定すると “Fn string too long” (n は指定したファンクションキーナンバー) のエラーとなります。

KEYのON、OFFの指定はコマンド行の最初のアーギュメントで行い、省略するとコマンド実行以前の設定となります。

“/I” オプションはファンクションキーをBASICの初期設定に再設定します。

また、文字列中にツールコマンドのアーギュメントの区切として使われる文字 (スペース、タブ、‘.’、‘+’、‘<’、‘>’、‘\’、‘/’、‘;’、‘|’) を指定したい場合、“DIR /W” などのようにダブルクォーテーションで囲みます。

例

```
A>key off 1, "DIR/W¥OD" ↵
```

(ファンクションキー1にDIR/Wと改行コードを設定)

LIST

書式

LIST [/H] □ 〈ファイル名〉

機能

BASICの中間言語ファイルをソースファイルに変換し出力します。

解説

DOS上でBASICのプログラムリストを見ることが出来ます。

入力ファイルには〈ファイル名〉により必ずBASICのテキスト（中間言語）ファイルを指定してください。それ以外のファイルを指定すると、

“〈ファイル名〉 is not a basic binary file”
のエラーとなります。

例

```
A>list TEST.BAS ␣  
100 CLEAR 1024 , &HD7FF -----  
110 FOR L=0 TO -----  
      ⋮  
150 END  
A>
```


書式

LS [/H] [/L] [/A] [/B] [/C] [/D] [/S | /T]
 [/R] [] [(ファイルスペック) | (ドライブ名)]

機能

ディレクトリの詳細情報を報告します。

解説

オプションを指定しない場合、カレントドライブのディレクトリ下のファイル名をアルファベット順にソートし、スクリーン幅に合わせて改行し表示します。

“/L” オプションにより以下の情報を付加し、各ファイルごとに改行して表示します。

- ・最新更新日時
- ・ファイルサイズ

以下のスイッチにより表示の内容、形式を変更することが出来ます。

- ・ “/A” 一行に一ファイルずつファイル名とファイル属性 を表示
- ・ “/B” “ ” ファイルサイズ “ ”
- ・ “/D” “ ” ファイルの日付 “ ”
- ・ “/C” 表示の形式を一行に一ファイルにする

ファイルの表示順序は以下のスイッチにより変更出来ます。

- ・ “/T” 最新更新日の新しい順に表示
- ・ “/S” ファイルサイズの大きい順に表示
- ・ “/R” 上記2つのスイッチとそれぞれ組み合わせることにより、各々逆に表示します。単独の場合、アルファベット順の逆順となります。

尚、“/T” “/S” が共に指定された場合 “/T” のみが有効となり日付順に表示されます。

例

A>ls /b/d/t b: 』 (日付の新しい順にソート)

ATEXT.DOC 1987-01-23 12:30 232345Bytes

BTEXT.DOC 1987-01-22 10:00 23358Bytes

CTEXT.DOC 1986-12-24 11:30 345112Bytes

ZTEXT.DOC 1985-09-15 12:30 2212Bytes

MENU

書式

MENU [/H] [/T|/S] [/R] _ [〈ファイルスペック〉 | 〈ドライブ名〉]

機能

ディスク上のファイルについて、連続してデリート、リネーム、タイプ、ダンプなどの処理を行うことが可能です。

解説

“MENU”と入力すると、カレントドライブのディスク上のファイル名が、アルファベット順にプロンプト表示されます。下記のサブコマンドにより各ファイルにつきコピー、リネーム等様々な処理を行うことができます。

またカレントドライブについてのみでなくドライブ名やファイルスペックの指定により他のドライブや特定ファイルの指定が可能です。

以下各コマンド操作について解説します。

- ・ ESC or “Q”, ^Z or ^C : コマンド処理の中止
- ・ SPACE or 『 』 : 現在表示されているファイルについて何も処理しません。次のファイルについて処理命令待ちの状態になります。
- ・ “E” : 表示中のファイルを消去します。
- ・ “R” : 表示中のファイルをリネームします。新しいファイルネームをひき続き入力します。
- ・ “T” : ファイルの先頭の10行をタイプアウトします。
- ・ “D” : ファイルの先頭から1画面分のダンプリストを表示します。
- ・ “C” : ファイルを他のファイルにコピーします。“C”を入力後、ひき続きコピー先の〈ファイル名〉を入力します。
- ・ BS : 1つ前のファイルにプロンプトが戻ります。

“T”及び“D”を実行した場合、プロンプトのファイル名は実行前のものがひき続き表示されます。従って、ファイルの内容を確認した後、デリート、等の処理を行うことが可能です。

ファイルの表示順序は以下のスイッチにより変更できます。

“/T” ファイルの日付の新しい順に表示

“/S” ファイルサイズの大きい順に表示

“/R” “/S” “/T”と組合せてそれぞれの逆に表示、単独ではアルファベットの逆順に表示します。

“/T” “/S”を同時に指定した場合“/S”は無視され日付順に表示されます。

例

A>menu

COMMAND.COM:

MSXDOS.SYS:

TEST.C:Rename

NEW FILE NAME :TEST1.C

Complete!

TEST2.C: Quit

A>

MORE

書式

MORE [/H] [/P 〈行数〉] _ [<ファイルスペック>]

機能

標準入力から標準出力へ1画面分ずつ区切りながら表示します。

解説

標準入力からの、または指定のファイルの内容を標準出力（通常はCRTディスプレイ）に1画面分表示します。入力がさらに続く場合、画面最下部に

“====MORE?====”

と表示されキー入力待ちの状態となります。

さらに、スペースキーをおすことにより、次の1画面分、リターンキーをおすことにより次の1行が表示され、再度キー入力待ちとなります。

以下、入力（ファイルまたは標準入力）の終りまで同様の操作により表示することが出来ます。

また、

“/P 〈行数〉”により1画面の出力行数を任意にかえることが出来ます。

“Q”（Quit）キーを押すことによりコマンドは中止されます。

“S”キーにより、現在表示中のファイルから次のファイルにスキップして出力します。入力が標準入力の場合や1ファイルのみ指定の時などはコマンドの終了となります。

例

A> MORE TEST.DOC ↵

THIS IS TEST FILE FOR MORE COMMAND

⋮

==== MORE ?====

この状態でスペースKeyをおすことにより
出力が再開されます。

PATCH

書式

PATCH [/H] [/S 〈ダンプ開始バイト〉] _ 〈ファイル名〉

機能

バイナリファイルを更新します。

解説

コマンド行の入力により指定のファイルがスクリーンに16進で表示され、パッチモード（ファイル更新モード）となります。カーソルを自由に動かしスクリーン上の数値を変更することにより、指定のバイナリファイルを更新することができます。ダンプ開始バイトは“/S”オプションにより4桁以内の16進数で指定可能です。指定しない場合ファイルの先頭（0バイト）からとなります。

さらにESCキーによりコマンドモードに入り、ファイルの任意のバイトを指定し更新することが可能です。以下各モードで使用できるキー、コマンドについて解説します。

パッチモード

“A～F” “1～9” の各キー

更新したいバイトの16進表示上にカーソルを移動し各キーを入力することによりメモリー上のファイルイメージを変更します。

“P”、“N”

それぞれ1画面分前（Pキー）、後（Nキー）へダンプ表示を進めます。

カーソルキー

カーソルを任意のバイト上に移動します。

スペースキー

カーソルを右に一つ移動。

バックスペースキー

カーソルを左に一つ移動。

ESCキー または、“\$”

コマンドモードへ移行。

コマンドモード

“P” ␣

パッチモードに戻ります。

“X” ␣

更新したメモリー上のファイルをセーブしてコマンドを終了します。

“Q” ␣

更新したメモリー上のファイルをセーブせずにコマンドを終了します。

〈ダンプ開始バイト〉の指定

ダンプ開始バイトを16進数4桁以内で指定しリターンキーを押すことにより指定のバイトからダンプ表示されパッチモードに戻ります。

SLEEP

書式

SLEEP [/H] [/A] [/B] _ [[<hh> ,] <mm> ,] <ss>

機能

システムを一時休止状態にします。

解説

SLEEPコマンドではシステムの再開時を時 (hh)、分 (mm)、秒 (ss) で指定しますが、指定の時間はデフォルトでシステム再開までのインターバル時間となり “/A” オプション指定時には時刻、すなわち hh 時 mm 分 ss 秒までのシステム休止となります。時、及び分は省略可能ですが、分のみの省略や “/A” オプション指定時の時、分両方の省略は無効です。またこの場合、時のデフォルトは当時刻となります。

“/B” スイッチにより、システムの再開時に BEEP 音を発生し、アラームとして用いることができます。

SLEEPコマンドは 1 / 60 秒に 1 回の割り込みをカウントすることにより時間を計っているため内部にクロックを持たない MSX でも動作することができますが、この場合 “/A” を指定した場合でも時刻の指定とはならず、インターバル時間として処理されます。

例

```
A>sleep /b 20  ␣
```

```
      ⋮  
      BEEP !
```

```
A>
```

SORT

書式

`SORT [/H] [/A] [/R] [/S <桁数>] _ [<ファイルスペック>]`

機能

文字列データをソート（並べ替え）します。

解説

入出力は通常標準入出力が用いられますが、入力については〈ファイルスペック〉でファイルの指定が可能です。

入力されたアスキーコードのデータを各行の一文字目からを対象に判断し、各行を特殊文字 1. 2. 3. ..., A. B. C. ..., ア. イ. ウ. エ, 順に並べ替えをします。アルファベットはアルファベット順にソートされた後、大文字、小文字の順に出力されます。

“/R” オプションをつけることにより上記の逆順にソートします。

“/S” オプションでは各行の何桁目からを判断の対象とするかを桁数で指定します。TAB は相当数のスペースとして数えます。

2つのオプションは同時に指定出来ます。

“/A” オプションによりアスキーコード順のソートをします。

“/R” と “/A” を同時に指定した場合、アスキーコード順の逆ソートとなります。

例

```
A>sort test.dat
1 2 3
4 5 6
A B C
a b c
B B B
b b b
アイウ
エオカ
A>
```

TAIL

書式

TAIL [/H] [/N] [/S 〈開始行〉 | /L 〈出力行数〉]
[〈ファイルスペック〉]

機能

入力から末尾から指定された行を切り取って出力します。

解説

入出力は通常標準入出力が用いられますが、入力については〈ファイルスペック〉で1つまたは複数のファイルが指定可能です。

入力の末尾から指定の長さを切り取り、行番号、入力がファイルの場合、ファイル名をつけて出力します。出力の開始位置は以下のスイッチにより指定します。

“/S” 出力の開始行を〈開始行〉で指定します。

“/L” 末尾の何行を指定するかを〈出力行数〉で指定します。

以上どちらも用いない場合、末尾10行の出力となります。

“/N” スイッチを指定すると、行番号、ファイル名が省かれ、行本体のみの出力となります。

例

```
A>tail b:test.doc 𐀀
      B:TEST.DOC
1225: line 1225
      ⋮
1234: line 1234
A>
```

書式

TR [/H] [/W] [/D] [/S] [/I] [/C] [/N] _
 “〈文字列1〉” _ [“〈文字列2〉”] _ [<ファイルスペック>]

機能

文字、文字列の置き換えを行います。

解説

TRコマンドでは、入出力は標準入出力ですが、<ファイルスペック>によりひとつまたは複数のファイルを指定可能です。また、“/W”スイッチにより、2種類の置き換えが選択可能です。“/W”を指定しない場合、文字単位の置き換えを行います。〈文字列1〉に含まれる文字が入力中に見つかると、その文字に対応する〈文字列2〉の文字に置き換えます。

また、“/W”スイッチを指定しない時にのみ、“/C”スイッチが使用出来ます。“/C”スイッチは〈文字列1〉に含まれない入力中の文字を〈文字列2〉に置き換えます。

“/W”を指定した場合、単語単位での置き換えを行います。〈文字列1〉と完全に一致する文字列が入力中に検出されると、〈文字列2〉に置き換え出力します。

①文字列の指定について

- ・文字列はダブルコーテーションで囲みます。
- ・〈文字列1〉は省略出来ません。また“/D”スイッチ指定時には〈文字列2〉は指定できません。
- ・文字列は255文字以下で指定し途中でヌルキャラクターを含んではいけません。
- ・次の三つのキャラクターは文字列中で機能コードとして特別の意味をもちます。

‘-’ (ハイフン)

二つの文字または数字を‘-’でつなぐことにより文字または数字の範囲の指定となります。

例 “I-N”と指定すると“I J K L M N”の指定と同等となります。

‘*’ (アスタリスク)

文字*N (Nは数字)で同一文字N個の文字列を意味します。

例 “Z*5”は“Z Z Z Z Z”と同等となります。

‘¥’ (円記号)

タブやキャリッジリターンなどの特殊コードやハイフン、アスタリスクなどを文字列の要素として指定したい場合‘¥’の後に以下のような指定をす

ることにより可能となります。

“¥n”	: ラインフィード (0Ah)
“¥r”	: キャリッジリターン (0Dh)
“¥t”	: タブコード (09h)
“¥x〈16進2桁〉”	: ‘x’ の後に2桁の16進数でコードを指定
“¥〈8進3桁〉”	: 3桁の8進数で指定 例. “¥102” は“A”
“¥〈文字〉”	: 円記号の後の文字はそのまま文字列の要素となります。たとえば“¥¥”のように円記号を文字列に指定可能です。

②オプションスイッチについて

TRコマンドには以下のようなオプションスイッチが指定可能です。

- “/H” ヘルプメッセージを出力します。
- “/W” 単語単位での置換 (デフォルトは文字単位) をします。
- “/C” 〈文字列1〉に含まれない文字を置換します。(“/W” 指定時は無効)
- “/N” ファイル名を出力から省きます。
- “/D” 〈文字列1〉の文字または語を入力中から削除して出力します。
- “/S” 同一の置き換えが連続する場合、1回の置き換えにまとめて出力します。
(“/W”、“/D” のどちらかが指定されている場合 “/S” は無効)
- “/I” 複数の連続したスペース、タブを1スペースとしてマッチングを行います。

③エラーメッセージ、警告メッセージ

TRコマンドでは文字列、オプションの指定が多様であるため発生する様々なエラーにユーザーが適切に対応可能なよう以下のようなメッセージを出力します。

・エラーメッセージ

コマンド処理の続行不可能なエラーであり処理を中止します。

“Illegal use of ‘-’”

文字列指定のための ‘-’ の使用法に誤りがあります。(‘-’ の前後に文字がなかったり同じ文字である、など)

“Illegal use of ‘*’”

‘*’ の前に文字がない、後に数字がないなどのエラー。

“String2 not necessary”

“/D” 指定時に、文字列2は指定出来ません。

`"Missing string#"`

String #の指定がありません。(＃は1または2)

・警告メッセージ

メッセージの後何らかのコマンド処理を実行します。

`"Warning: multi definition '?'`

キャラクタ?が文字列中に2回以上指定されています('?' は実際指定された文字が表示されます)。最初の出現時の対応で置き換えをおこないます。


`"Warning: string2 is longer
Rest of string2 is ignored"`

単語単位の置き換えのとき文字列2が長く文字列1中に対応する文字が存在しません。文字列2の長すぎる部分を無視して置き換えを行います。

`"Warning: string1 is longer
Rest of string1 will deleted"`

単語単位の置き換えで文字列1が長すぎ文字列2中に対応する文字が存在しません。文字列1の長すぎる部分に対応する入力中の文字は削除されます。

例

A> TR "a-z" "A-Z" text.c 

(ファイルTEXT. Cのアルファベットの小文字を大文字に置き換えコンソールに出力します。)

A>TR /W /N "printf(" "fprintf(stderr" CTEXT.C >CTEXT1.C 

(置き換えの結果をファイルにリダイレクトする場合)

UNIQ

書式

UNIQ [/H] [/N] [/C] [〈ファイルスペック〉]

機能

入力中の隣接する 2 行を比較し、それが同じである場合、2 行目を出力しません。

解説

入出力は標準入出力が用いられます。

UNIQ の機能は SORT との組合せで特に有効となります。SORT の結果を UNIQ フィルターにかける事により、同一行の連続出力を避けることができ、見易い出力になるでしょう。

“/C” スイッチにより行頭にその行の出現回数が表示されます。

“/N” スイッチによりファイル名、行の出現回数を省きます。

例

```
A>sort <file1 >file2
```

```
A>uniq <file2 >list1
```

(以上のオペレーションにより `f i l e 1` の各行が辞書順にソートされ同一内容の行が重複しないリストが `l i s t 1` としてディスク上につくられました。)

VIEW

書式

VIEW [/H] _ 〈ファイル名〉

機能

ファイルの内容を画面表示し、カーソルキー、その他のキーにより画面外も自由に見ることができます。

解説

“VIEW” コマンドを実行すると、〈ファイル名〉で指定されたファイルが先頭から1画面分表示されます。以下カーソルキーの操作により、画面を上下自由にスクロールすることができ、1画面に納まらないファイルの内容をすべて見る事が可能です。

カーソルアップキー	5行上を表示
ダウンキー	5行下を表示
ライトキー	8桁右を表示
レフトキー	8桁左を表示

カーソルキーの他に

“P”	前ページを表示
“N”	次ページを表示
“H”	ファイルの先頭を表示
“T”	ファイルの末尾を表示

の各キーにより表示部分を自由に移動できます。

また、“?” キーによりVIEWコマンドのサブコマンドのヘルプが表示されコマンドを起動中に参照することができます。

“Q”、“q”、“^C”又はESCキーによりVIEWコマンドから抜けられます。

WC

書式

WC [/H] [/P 〈行数〉] _ [〈ファイルスペック〉]

機能

入力中の語数、行数、ページ数をカウントします。

解説

入出力には通常標準入出力が用いられますが、入力には〈ファイルスペック〉により、1つまたは複数のファイルの指定が可能です。

[/P 〈行数〉] により1ページ毎の行数を指定出来ます。デフォルトは60行です。

WCコマンドにおいては、文字、数字のブランクを含まない列（但し、“ ” “-” 等が間にあっても連続とみなす）を1ワードとします。

それ以外のものは無視されます。

例

A>wc *.doc

	Pages	Lines	Words	Characters
A.DOC	12	116	441	1543
B.DOC	1	12	25	85
C.DOC	12	110	500	1834

A>

6 エラーメッセージ一覧

ツールコマンドではすべてのエラーメッセージはコンソール画面にのみ出力されます。プリンターやファイルにエラーメッセージが出力されることはありません。

`Bad argument (+usage)`

アーギュメントの数に誤りがあったり、不要なアーギュメントがコマンドライン中にあった場合このメッセージとともにユーセージを表示しコマンドを終了します。

`Bad field specified`

BODY、DUMPコマンドで出力の開始行（バイト）の指定が終了行（バイト）より大きい場合このメッセージで処理を終了します。

`Bad hexadecimal`

16進数の指定に誤りがあります。

例 `tr "☞0x" "0a"` (xは16進数として不適切なためエラーとなる)

`Excessive argument (+usage)`

アーギュメントの数が多すぎます。複数のファイル名を '+' または ',' でつながずに指定したような場合このメッセージが表示されます。

例 `ls *.c *.com *.rel`

`File cannot be copied onto itself`

MENUコマンドの "C" (コピー) コマンドでコピー先のファイル名がコピー元と同一に指定された場合このメッセージを表示します。コピーはおこなわれずプロンプトのファイル名はそのまま次のコマンド待ちとなります。

`<ファイル名> is not a hexfile`

BSAVEコマンドでHEXファイルでないファイルを入力したときにこのメッセージでコマンドを終了します。

Illegal combination of switch

オプションスイッチの指定に無効な組合せがあります。その後の処理はコマンドにより異なります。(BODY、MENU、LSの各解説を参照してください)

例 ls /s/t

Invalid month specified (+usage)

CALコマンドで1～12の範囲以外で月の指定があった場合のメッセージ。

Invalid character in '/x' (+usage)

オプションスイッチ中に不要なパラメータがあります。

例えば '/H' オプションなどのようにパラメータの不要なオプションにおいて '/H1' などのように無効な文字や数字が指定されてる場合ユーセージとともにこのメッセージを表示しコマンドを終了します。

Invalid key number (+usage)

KEYコマンドにおいて1～10以外のファンクションキーナンバーの指定があった場合このメッセージでコマンドを終了します。複数のキーの指定のなかでひとつでも誤りがあった場合このエラーとなり処理を中止します。

Invalid minute or second specified (+usage)

SLEEPコマンドで60以上の分または秒の指定があったときにこのエラーとなります。ただし、時のパラメータ省略時の分、時及び分のパラメータ省略時の秒の指定には60を超えた値も許されます。

missing [[hh] mm] ss (+usage)

SLEEPコマンドの時刻指定（/A指定時）で時、分のパラメータが共に省略されたときこのエラーとなります。

'x' is not digit (+usage)

数字でパラメータを指定しなければならないオプションスイッチで数字以外を指定した場合このエラーメッセージが表示されます。

例 HEAD /S12FG

の場合 “'F' is not digit” と表示されます。

Missing parameter (+usage)

HEADコマンドの '/L' オプションのように必ずパラメータが必要なところで指定がないとこのメッセージでエラー終了となります。

例 HEAD/L では “Missing parameter in '/L' ” となります。

Not enough memory for buffer

コマンドの実行に十分なメモリーを確保できなかった場合にこのメッセージを表示しコマンドを終了します。

File not found: <ファイル名>

指定のファイルがディレクトリ中に見つかりません。

Rename error

MENUコマンドの 'R'（リネーム）コマンドにおいて指定された新しいファイル名がすでにディスク上に存在するファイルと同一である場合にリネームエラーとなりプロンプトのファイル名はそのまま次のコマンド入力待ちとなります。

String too long (+usage)

TRコマンドで255文字を超える文字列の指定があった場合、またはKEYコマンド

のファンクションキーへの設定文字列が15文字を超えた場合このメッセージを表示しコマンドの実行を終了します。

Syntax error (+usage)

KEYコマンドで文字列設定の書式にまちがいがある場合のエラーメッセージです。

例 KEY F1. DIR

ではファンクションキーナンバーと設定文字列の区切りにカンマでなくピリオドが使われているためこのエラーとなります。

Unknown switch (+usage)

無効なオプションスイッチが指定された場合のエラーメッセージです。

例 WC /K

では、WCコマンドのオプションに/Kは存在しないためこのエラーとなりコマンドを終了します。

付録． MSX-DOS TOOLS 使用サンプル集

MSX-TOOLSの、2つ以上のコマンドを組み合せた例を、サンプルとしてここに挙げます。

ここでは、あらかじめフロッピーに入っている、

sample1.doc

sample2.doc

sample3.doc

および、MSX-DOS立ち上げ時に使われる、

welcome.doc

が、必要です。

また、ここで使用するフロッピーは、必ず、マスターからコピーしたものを、用いて下さい。また、ライト プロテクト ノッチはoff（書き込み可能）としておいて下さい。2つ以上のTOOLSのコマンドを用いた場合、コマンドからコマンドヘータを引き渡すのに、中間ファイルを作ります。従って、ライトプロテクトスイッチをon（書き込み不可）としておくと、中間ファイルが作れずにエラーとなってしまいます。

まず、サンプルとして付いてくるファイルの内容を見てみましょう。

A>type sample1.doc

と、タイプして見て下さい。

4
2
7
5
8
1
3
6

A>

と、表示されると思います。では、このファイルをソートしてみましょう。


```
A>sort sample1.doc
```

```
1
2
3
4
5
6
7
8
```

```
A>
```

では、もうちょっと複雑なことをしましょう。
いまソートしましたが、たとえばソートの結果のうち、最初から3つめまでを
表示させるには、次の様にします。

```
A>sort sample1.doc | head /L3
```

```
1: 1
2: 2
3: 3
```

```
A>
```

ここで出てきたheadコマンド（ファイルの先頭（はじめ）から、指定行だけ表示する。）について、もうすこし説明しますと、
まず、

```
A>dir *.doc
```

で、拡張子が「doc」のファイルを見てみます。おそらく、

```
WELCOME.DOC
SAMPLE1.DOC
SAMPLE2.DOC
SAMPLE3.DOC
```

の4つだと思います。そこで、たとえば

```
A>head /L3 welcome.doc
```

とすると、

```
1:
2:   ようこそ MSX-DOS/TOOLS の せかいへ
3:
```

のように表示されます。これは、このファイルのうへ（最初）から3行を表示したものです。では、つぎに、

```
A>head /L5 *.doc
```

としますと、これは このフロッピー上の（この場合はAドライブの）全てのファイルのうち、拡張子にdocを持つもののファイルの初めから5行を、順に表示します。

```
welcome.doc
```

```
1:
2:   ようこそ MSX-DOS/TOOLS の せかいへ
3:
4:   このたびは、MSX-TOOLSを おかいあげ
5:   いただき まことに ありがとう ございます。
```

```
sample1.doc
```

```
1: 4
2: 2
3: 7
4: 5
5: 8
```

```
sample2.doc
```

```
1: 1234567890
2: 2234567890
3: 3234567890
4: 4234567890
5: 5234567890
```

```
sample3.doc
```

```
1: 2222
2: 3333
```

```
3: 4444
4: 1111
5: 2222
```

A>

いまは、これだけ（4つのファイルだけ）ですが、もっとファイルが多くなって、1画面に入らずに、スクロールアップして画面の上に消えて行ってしまう場合、

A>head /L5 *.doc | more

とすれば、1画面1画面ごとにスクロールが止まって、見やすくなります。

さて、tail（ファイルの後ろ（おしまい）から指定行だけ表示する。）でも指定の仕方は同じです。

A>tail /L5 *.doc

で、該当するファイル（ここでは、sample1.doc、sample2.doc、sample3.doc、welcome.docの4つのはずです）のファイルのおわりから5行を順に表示します。

```
welcome.doc
14: と して あります. これらの コメント`を おつかい
15: の うえ, より たのしい MSX-LIFEを おすすめ
16: し ください.
17:
18:      -----ASCII MSXとうかつぶ
```

```
sample1.doc
4: 5
5: 8
6: 1
7: 3
8: 6
```

```
sample2.doc
4: 4234567890
5: 5234567890
6: 6234567890
```

7: 7234567890
8: 8234567890

sample3.doc
7: 3333
8: 6666
9: 1111
10: 8888
11: 7777

A>

これらのhead、tailコマンドは、「どのファイルだったかなー」というときに、非常に役に立ちます。

また、ファイルの最初の方に、ファイルを作った日にちや、バージョン、作者の名前などを書いておけば、headコマンドにより、すぐに見つけ出す事が出来ます。

ここまで読んでこられたあなたなら、次の例も容易に理解できるでしょう。

A>SORT SAMPLE3.DOC | HEAD /L4 | MORE

つぎに、同様にファイルの中を見るコマンドで、DUMPコマンドを使った例をあげましょう。

A>DUMP sample1.doc

0000 : 34 0D 0A 32 0D 0A 37 0D 0A 35 0D 0A 38 0D 0A 31 4..2..7..5..8...
0010 : 0D 0A 33 0D 0A 36 0D 0A 1A ..3..6..

A>

このように、ファイルの内容を16進とアスキーコードで表示します。上の例では、34が数字の4、0Dがキャリッジリターン（改行）、0Aがラインフィード（1行送り）...となっています。1Aはファイルの終わりを表します。そのため、つぎのバイトから先は表示されません。先ほどのHEADコマンドと組み合わせると、下のようになります。

A>DUMP sample1.doc | HEAD /L1

sample1.doc

1: 0000 : 34 0D 0A 32 0D 0A 37 0D 0A 35 0D 0A 38 0D 0A 31 4..2..7..5..8..1

A>

最後に、M80、L80、BASICを組み合わせて使用した例を示します。

ここでは、

```
SEARCH. MAC , M80. COM , LIST. COM
SEARCH. BAT , L80. COM , BSAVE. COM
SEARCH. BAS
```

の7つのファイルが必要となります。

これは、SEARCH. BATと言うバッチファイルを使って実行するようになっています。まず、使用するファイルの内容を見てみましょう。

A>TYPE SEARCH. BAT	動作の説明 (これはファイルには入ってません)
m80 =search	search.macをアセンブルする
l80 /p:d000,search,search/n/x/e	アセンブルした結果をリンカーに通す
bsave search.hex >search.bin	リンカーの出力をバイナリーファイルにする
basic search.bas	basic上でsearch.basを実行する

A>

最初の3行で、SEARCH. BINが作られ、basicプログラムのSEARCH. BASの中で、これがメモリーにロードされ、使われます。ですから、L80. COMを実行する際に、&hD000hからロードするように指定しています。(/p:d000の部分)

つぎに、SEARCH. BASの内容を見てみましょう。このSEARCH. BASは、BASIC上の中間言語ファイルとなっていますので、このままTYPEコマンドで見るとは出来ません。そこで、LISTコマンドを使って、内容を見ます。(ちなみに、中間言語ファイルの方が、ディスク上のスペースを取りません。)

A>LIST SEARCH. BAS

```
110 CLEAR 300,&HCFFF: BLOAD"SEARCH.BIN": DEF USR=&HD000
120 DIM A$(100): N=0: I=0: A$(N)=MID$(USR(CHR$(0)+"???????DOC"),2,8)
130 IF A$(N)<>"" THEN N=N+1: A$(N)=MID$(USR(""),2,8):GOTO 130
140 IF A$(I)="" THEN BEEP: END
```

```

150 GOSUB 170
160 I=I+1: GOTO 140
170 '
180 B$=A$(I)+"*.DOC": OPEN B$ FOR INPUT AS #1
190 PRINT SPC(8); " ファイルノ ナミハ:"; B$
200 IF EOF(1) THEN CLOSE: PRINT SPC(8); B$; " ノ ナイウワ ヒョウジ" シマシタ.": RETURN
210 INPUT #1, C$
220 PRINT C$
230 GOTO 200

```

A>

さて、SEARCH. BASも、説明しておきましょう。このプログラムは、2つの部分に分けられます。それは、110行から160行の部分と、170行から終りまでです。

このうち、110行から160行までがSEARCH. BINを使用した部分で、このBASICプログラムの重要な部分です。110行は、機械語のエリアを確保して、(&HDO00hから上)そこにSEARCH. BINをロードします。そしてSEARCH. BINの実行開始番地を&HDO00とします。

120行では、文字配列A\$を宣言し、SEARCH. BINに"????????DOC" (つまり"*.DOC")の指定をして、ディスクの中から最初に該当するファイル名をA\$(0)に代入しています。またIとNに0を代入して、初期設定をしています。

130行では、ディスクのファイル名の中から、"????????DOC"に該当するファイル名をすべて、文字配列A\$に順次代入します。ファイル名を得るために、SEARCH. BINを繰り返し実行しています。(USR(" ")) SEARCH. BINは" "を指定されると、後続のファイル名を順次返すようになっています。また該当するファイルがない場合には、" " (ヌルストリング)を返すようになっています。

この時点で、文字配列A\$の中は次のようになっています。

```

A$(0) = " WELCOME"
A$(1) = " SAMPLE 1"
A$(2) = " SAMPLE 2"
A$(3) = " SAMPLE 3"
A$(4) = " "

```

140行では、文字配列A\$の中身が" " (ヌルストリング)であれば、このプログラムを終了します。今ここを最初に通るときには、I=0ですから、A\$(0)=" WELCOME" となり、先へと進みます。I=4のときに、A\$(4)=" " となり、終了します。

150行は、170行以下のサブルーチンを呼んでいます(170行へいき、RETU

RN文で帰ってくる)

160行では、Iの値を1増やし、140行へと飛びます。140行と160行により、Iの値を0、1、2、...と増やして、つまり文字配列A\$のファイル名をつぎつぎと変えながら実行していくわけです。

170行以降は、まとめて説明しますと、B\$に文字配列A\$からファイル名を代入して、そのファイルをシーケンシャルファイルとしてオープンし、ファイルの内容を画面に表示して、表示する内容が無くなったらRETURN文により150行に帰る、というものです。

従って、120行の"????????DOC"と、170行以降を変える事により、「ディスクのファイル名を得て、それを元にして何かする」というプログラムが簡単に出来るのです。

では、実行です。と言っても、ただ見ているだけで、アセンブルからBASICの実行まで、全部バッチファイルがやってくれます。

お買い上げ頂いたマスターフロッピーからのコピーをお使いください。

A>SEARCH (SEARCH.BATを実行)

A>m80 =search

No Fatal error(s)

A>l80 /p:d000,search,search/n/x/e

MSX.L-80 1.00 01-Apr-85 (C) 1981, 1985 Microsoft

Data D000 D0A7 < 167>

43403 Bytes Free

[0000 D0A7 208]

A>bsave search.hex >search.bin
complete

A>basic search.bas

(ここで画面が消えて、BASICが立ち上がる)

SEARCH.BASを実行する。

どうでしたか。あまりにすいすいと実行してしまうので、よくわからなかったと思います。そのときは、CALL SYSTEMとタイプして、MSX-DOSにもどり、もう一度SEARCH.BATを実行してみてください。また、SEARCH.BASについて、もっと詳しく動作を追って見たい方は、BASIC上でTRONコマンドを実行した上で、RUNしてみてください。SEARCH.BINはもう作られていますので、今度は、

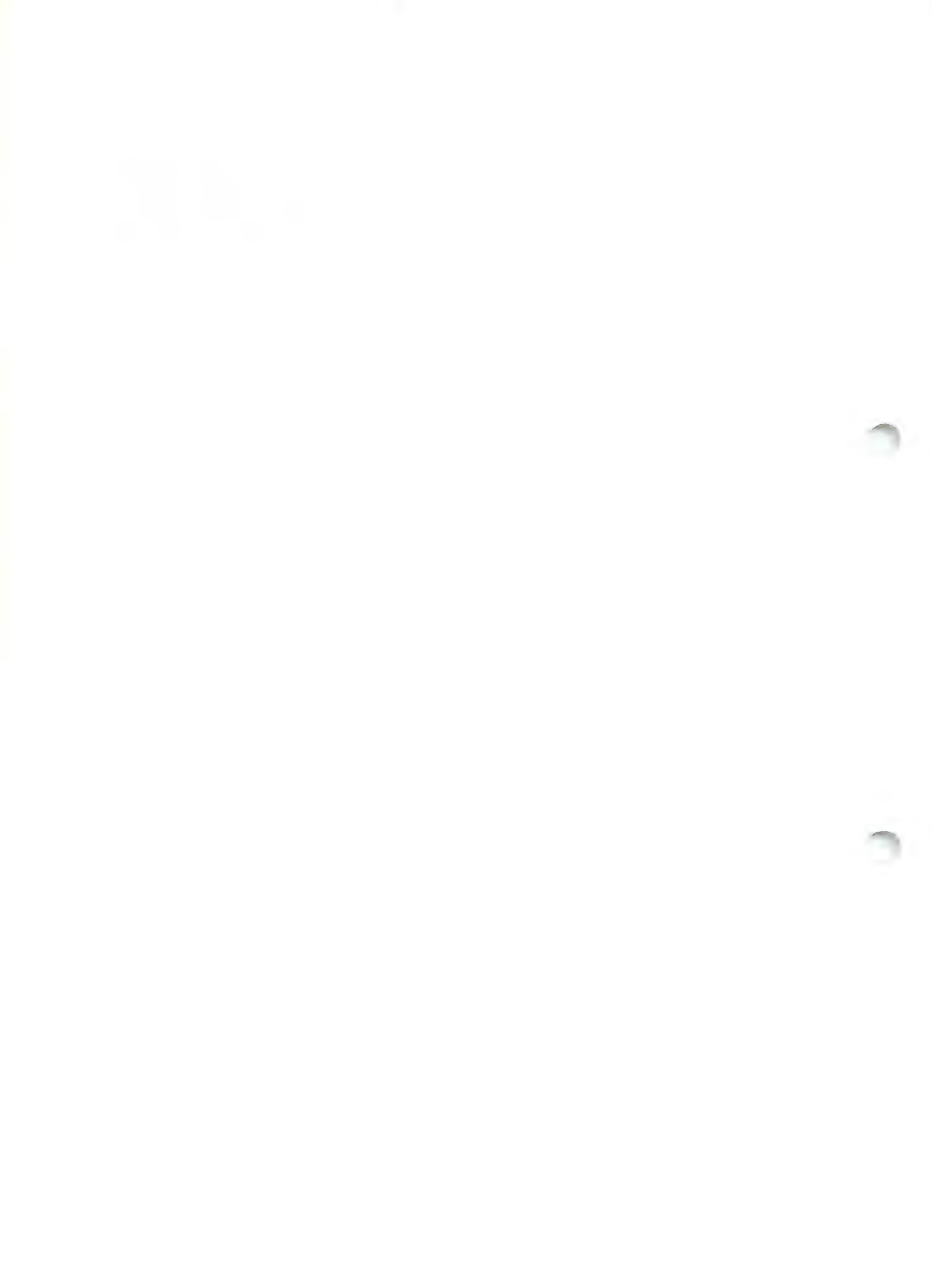
A>BASIC

(BASICが立ち上がる)

LOAD" SEARCH.BAS"

.
.
.
.

から 始めることができます。



MSX

MSX-DOS MANUAL

copyright (C) 1987 , Ascii Corporation

はじめに

本書は、MSXコンピュータのためにマイクロソフト社が開発したディスク・オペレーティング・システムMSX-DOSの解説書です。

MSXコンピュータは、1983年に(株)アスキーとマイクロソフト社が提唱したコンピュータの統一仕様です。MSXは、優れた拡張性を持つことや、周辺機器やソフトウェアの互換性があることなどが高く評価され、全世界で多くの人々に愛用されています。また、1985年には、MSX仕様をさらに拡張したMSX2統一仕様が発表されました。

MSXは、ディスクを使用することで多くのプログラミング言語やアプリケーション・プログラムが利用できるようになります。またMSX2は優れた映像機能を持っていますが、その機能をより発揮するにはやはりディスクの利用が不可欠です。このようにディスクの必要性が高くなると、ディスクそのものをコントロールするソフトウェアが要求されるようになります。MSX-DOSは、この要求に応じて開発された、MSXのためのディスクをコントロールするソフトウェア(ディスク・オペレーティング・システム)なのです。

MSX-DOSは、MS-DOSに準じた操作性を備えており、またファイルの互換性があります。そしてCP/Mのプログラムを容易に移植できるなどの、数々の特徴を持っています。そのためMSX-DOSを利用することで、MSXコンピュータをより高度なコンピュータ・システムとして活用することが可能になるのです。

本書が、あなたのMSXの世界をより広げることに役立てばと、期待してやみません。

本書は、「第1部 MSX-DOSイントロダクション」、「第2部 MSX-DOSユーザーズ・ガイド」、「第3部 MSX-DOSプログラマーズ・ガイド」の3部構成になっています。

第1部 MSX-DOSイントロダクション

MSX-DOSを使用する上で必要となる基本的知識について述べています。初めてディスク・オペレーティング・システムに接する方は、第1部を読むことによりMSX-DOSの概要を理解することができます。

第2部 MSX-DOSユーザーズ・ガイド

MSX-DOSの基本的な操作方法やコマンドについて解説しています。プログラミング言語やアプリケーション・プログラムを利用する方は、第2部を読むことによりMSX-DOSの操作法を理解することができます。

第3部 MSX-DOSプログラマーズ・ガイド

MSX-DOSのシステム・コールについて解説しています。アセンブラやコンパイラを使ってアプリケーション・プログラムを開発する方は、第3部を読むことによりMSX-DOSの活用法を理解することができます。

なお各部ごとに用語解説を付けましたので、必要に応じて参照してください。

1987年2月

(株)アスキー MSX統括部

第1部
イントロダクション



はじめに

第1部イントロダクションは次のように構成されています。

第1章では、初めてオペレーティング・システムに接する方のために、オペレーティング・システムについて簡単に解説しています。

第2章では、最新のオペレーティング・システムであるMSX-DOSの数々の優れた特徴を紹介しています。

第3章では、MSX-DOSの使用例を紹介しています。

第4章では、MSX-DOSのもとでどのようにしてプログラミングを行うかについて、アセンブラ言語を用いた場合の例を上げて解説しています。

付録Aでは、MSX-DOSを使用するために必要なハードウェアとソフトウェアについて述べています。

付録Bでは、MSX-DOSのもとで利用できるアプリケーション・プログラムを紹介しています。

付録Cでは、MSX-DOSについての知識をより深めたい方のために、参考となる書籍を紹介しています。

第1章	オペレーティング・システム	1
第2章	MSX-DOSの特徴	2
第3章	MSX-DOSの使用例	4
第4章	MSX-DOSでのプログラミング例	5
付録A	MSX-DOSのハードウェア、ソフトウェア構成	9
	A. 1 ハードウェア構成	
	A. 2 ソフトウェア構成	
	A. 3 MSX-DOS概念図	
付録B	アプリケーション・プログラムの紹介	11
付録C	関連書籍の紹介	12
※用語解説		13
※索引		16

第 1 章 オペレーティング・システム

マイクロ・コンピュータが誕生したころ、まだその機能は低く周辺機器も少かったため、走るプログラムも単純なものでした。しかし年々マイクロ・コンピュータの心臓部である中央処理装置（CPU）の性能は向上し、内部記憶装置（メモリ）の容量も増大したことにより、周辺機器の性能は上がり種類も増えてきました。そしてこれにともなって、それを利用するプログラムも多様化し、大量のデータを扱うものも出てきました。

このように進歩したコンピュータの機能を充分に活用するには、周辺機器を含めたコンピュータ・システムそのものをコントロールすることが必要です。また多数のプログラムや大量のデータを管理するための手段も不可欠になりました。そして、このようなことは人が行うよりも、コンピュータ自身に行わせた方が効率的である、ということがわかってきました。こうして生まれたソフトウェアがオペレーティング・システム（OS）です。

さて、進歩を重ねてきたコンピュータ・システムの中でも外部記憶装置の進歩は著しく、短い月日の間に紙テープからカセットテープ、さらにはフロッピー・ディスクへと長足の進歩を遂げました。フロッピー・ディスクは、1枚に多種多様なプログラムやデータ（ファイル）を記録することができ、入出力も高速なので、今日のコンピュータ・システムには欠かすことのできないものとなっています。

このフロッピー・ディスクの持つ特徴を生かして設計されたOSをディスク・オペレーティング・システム（DOS）といい、本書で解説するMSX-DOSもそのひとつです。ディスクに記録されるファイルは、多いときで100以上になります。DOSの主な仕事はこのディスク・ファイルを管理することです。DOSには、ファイルに関する情報の表示、あるいはファイルの削除、ファイル名の変更など、ファイルを管理するための有用な命令（コマンド）がいくつか備えられています。またディスクに対する入出力もDOSが行うので、人はどのようにディスクを扱うか、どのようにファイルを扱うかということは意識する必要がありません。そのため、プログラミングなどの作業をより効率的に行うことができるのです。

つまりOSとは、コンピュータの使用環境を整え、コンピュータ資源を有効にかつ効率良く利用するための、基本的なソフトウェアということができます。

第2章 MSX-DOSの特徴

MSX-DOSは、8ビット・マイクロプロセッサ用のOSとして、数々の優れた特徴を備えています。以下にその特徴について解説します。

i. MS-DOSに準じた高度な操作性

MSX-DOSは、16ビット・マイクロプロセッサ用の汎用OSであるMS-DOSをもとに開発されました。そのためMSX-DOSを知ること、MS-DOSの操作にも慣れることができます。

- ① MSX-DOSのコマンドの名前と機能はMS-DOSと共通になっています。そのため、MSX-DOSを使用することで、MS-DOSの使用法を学ぶことができます。
- ② ファイルを修正すると、その日付と時刻が自動的にディスクに記録されるので、いつファイルが修正されたかを知ることができます。
- ③ コマンド行の編集機能を備えています。そのため入力ミスを容易に修正することができ、また同じコマンドを続けて入力したり、似たコマンドを入力したりするのが楽になっています。
- ④ MSX-DOSは、実行するプログラム名を“autoexec.bat”というファイルに書き込んでおくだけで、起動時に自動的に実行する機能を備えています。

ii. MS-DOSとファイルの互換性がある

ディスクへの記録方法がMS-DOSと同一なので、MS-DOSで作成したファイルをMSX-DOSで読み書きすることも、その逆も可能です。

iii. 強力、高速かつ柔軟なファイル処理

MSX-DOSは以下のような優れたファイル処理を実現しています。

- ① MS-DOSやUNIXと同様に最大1ギガバイトという大容量のファイルを取り扱うことができます。

- ② F A Tやディレクトリのバッファリングにより、高速なディスクの入出力を実現しています。またブロック入出力を利用したプログラムでは、さらに高速な入出力が可能です。
- ③ ファイルの処理に論理レコードと呼ばれる手法を取り入れて、プログラムの開発を容易にし、またディスクの有効利用を図っています。
- ④ 周辺機器をディスク・ファイルと同じように取り扱うことができます。たとえば、ファイルからの入力と同じ方法でキーボードから入力したり、ファイルに出力するようにしてプリンタに出力したりすることができます。

iv. D i s k B A S I CをD O Sから利用できる

M S X D i s k B A S I CはM S X-D O Sを利用して動作しています。そのため、B A S I Cの画面編集機能や画面制御命令などの便利な機能を、M S X-D O Sから利用することができます。

- ① ファイル処理に同じ方法を用いているので、D i s k B A S I Cで作成したファイルをM S X-D O Sで扱うことも、その逆も可能です。
- ② M S X-D O SにはD i s k B A S I Cに移行するコマンドが、またD i s k B A S I CにはM S X-D O Sに移行するコマンドが用意されているので、相互に行き来することができます。

v. フロッピー・ディスク・ドライブが1台で動作する

他のO Sは、動作するために最低でも2台のフロッピー・ディスク・ドライブを必要とするのが普通です。しかしM S X-D O Sは1台でも動作するように設計されているので、特にその必要を認めない方は、2台目のフロッピー・ディスク・ドライブを設置しなくても、M S X-D O Sを利用することができます。

vi. C P / Mからのプログラムの移植が容易

M S X-D O Sのシステム・コールの大部分は、同じ8ビット・マイクロプロセッサ用のO SであるC P / Mのファンクション・コールと互換性を持っています。したがって、C P / M上の多くのプログラムを移植してそのまま利用することができます。

第3章 MSX-DOSの使用例

MSX-DOSにはコンピュータの制御やファイルの処理を行ういろいろなコマンドが備えられていて、その名前を入力するだけですぐにその処理が実行されるようになっています。

コマンドのなかでも最もよく使われるものが“dir”です。“dir”コマンドはファイルに関する情報を表示します。MSX-DOSではファイル名、ファイルの大きさ、最後に修正された日時などが表示されます。以下にその例を示します。

```
A>dir  ← “dir” コマンドを入力する
MSXDOS   SYS       2432  85-08-23   9:29p
COMMAND  COM       6656  85-09-02  10:10p
AUTOEXEC BAT       128   85-05-18   4:30p
TEST1    COM      34944  85-10-23  11:15a
SAMPLE   COM       128   85-09-10   3:10p
↑ ファイル名           ↑      : ↑修正した日付   ↑ 時刻
                        ファイルの大きさ(バイト) :
                        :
SAMPLE     MAC       256  85-09-10   3:07p
SAMPLE     PRN      1280  85-09-10   3:07p
           66 files      22528 bytes free
A>        ↑表示したファイルの数   ↑ディスクの空き領域
           へ次のコマンドの入力を待っている
```

“dir”コマンドを実行すると、ディスクに入っているファイルが記録された順に表示されます。

この他に、ファイルの内容を表示する“type”や、ファイルを削除する“del”、ファイル名を変更する“ren”などのコマンドがあります。

またディスクに記録されている外部コマンドやプログラム、バッチ・コマンドなども、コマンドと同様にその名前を入力するだけで、すぐに実行することができます。

第4章 MSX-DOSでの

プログラミング例

MSX-DOSのもとで動作するプログラムは、基本的には機械語で、普通「実行ファイル」とか「コマンド・ファイル」などと呼ばれています。

MSX-DOSには、機械語プログラムで入出力やファイルの操作などを行うために、「システム・コール」というものが備えられています。機械語プログラムは、アセンブラやコンパイラを使って、このシステム・コールを利用して開発することになります。

次に、“Welcome to MSX-DOS Programming world !!”というメッセージをディスプレイに表示する、簡単なアセンブラによるプログラム例を示します。

```
title Sample program for MSX-DOS
;
        . z80
;
start:
    ld     de, msg    ←メッセージ・データの先頭アドレスを
                        deレジスタにセット
    ld     c, 9        ←機能番号9をcレジスタにセット
                        (ディスプレイにデータを表示)
    call   5           ←システム・コールを実行
                        (5番地をコール)
    ret
msg:
    db     'Welcome to MSX-DOS '
    db     'Programming world !!'
    db     '$'        ←“$”の前までが表示される
;
    end
```

このプログラムをアセンブルすると、機械語のコマンド・ファイルが作られます。

図4.1に機械語プログラムの開発手順を示します。

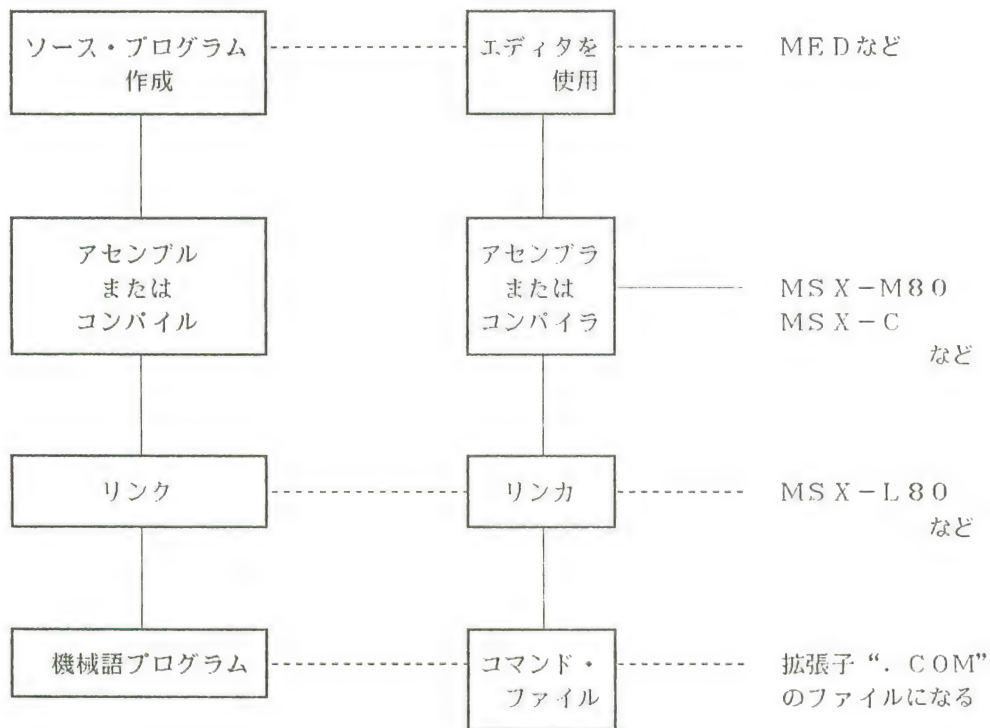


図4.1 機械語プログラムの開発手順

またシステム・コールの他の機能番号を用いると、ディスクに対する入出力やファイルの管理などを行うプログラムを作ることができます。システム・コール一覧を表4.1に示します。

表4.1 システム・コール一覧

番号	機 能	番号	機 能
0	システム・リセット	1	コンソール 1 文字入力
2	コンソール 1 文字出力	3	補助入力装置 1 文字入力
4	補助出力装置 1 文字出力	5	プリンタ 1 文字出力
6	コンソール直接入出力	7	コンソール直接 1 文字入力
8	コンソール直接 1 文字入力 ¹⁾	9	文字列出力
10	1 行入力	11	コンソール入力状態チェック
12	バージョン番号読み出し	13	ディスク・リセット
14	標準ドライブ選択	15	ファイル・オープン
16	ファイル・クローズ	17	ファイル・サーチ
18	ファイル・サーチ ²⁾	19	ファイル削除
20	シーケンシャル・リード	21	シーケンシャル・ライト
22	ファイル作成	23	ファイル名変更
24	オンライン・ドライブ情報読み出し	25	標準ドライブ番号読み出し ³⁾
26	DMAアドレスのセット	27	ドライブ・マップ情報読み出し
28	機能無し	29	機能無し
30	機能無し	31	機能無し

32	機能無し	33	ランダム・リード
34	ランダム・ライト	35	ファイル・サイズの読み出し
36	レコード番号のセット	37	機能無し
38	ランダム・ブロック・ライト	39	ランダム・ブロック・リード
40	ランダム・ライト ^{*4)}	41	機能無し
42	日付の読み出し	43	日付のセット
44	時刻の読み出し	45	時刻のセット
46	書き込み検査の設定	47	ディスクの直接読み出し
48	ディスクへの直接書き込み		

- *1) CTRL+C、CTRL+P、CTRL+N、CTRL+Sなどの特殊文字の入力のチェックを行います。
- *2) 条件を満たすファイルが複数ある時に使用し、次のディレクトリからファイルを探します。
- *3) 普通、機能番号14のシステム・コールとともに使用されます。
- *4) ディスクへの書き込みを行った後、データを書かなかったレコードを0で満たします。

付録A MSX-DOSの ハードウェア、ソフトウェア構成

A. 1 ハードウェア構成

MSX-DOSが動作するハードウェアの構成は次のとおりです。

i. 必要なもの

※64Kバイト以上のメインRAMを実装したMSXあるいはMSX2コンピュータ
ただし、RAMが64Kバイト未満のMSXコンピュータでも、増設することで動作します。

※ディスプレイ

※1台以上のフロッピー・ディスク・ドライブ

ii. あると便利なもの

※プリンタ

A. 2 ソフトウェア構成

MSX-DOSは以下のソフトウェアによって構成されています。これらのソフトウェアは、ディスク・インターフェイス・カートリッジおよびシステム・ディスクに含まれています。

①ディスク ROM

ディスク装置とともに供給されるディスク・インターフェイス・カートリッジ内のROMには、ディスクをはじめとする周辺機器との入出力を行うためのソフトウェアが記録されています。

②MSXDOS.SYS

MSXDOS.SYSは、MSX-DOSの中心で、システム・コールの実行やエラー処理を受け持ちます。

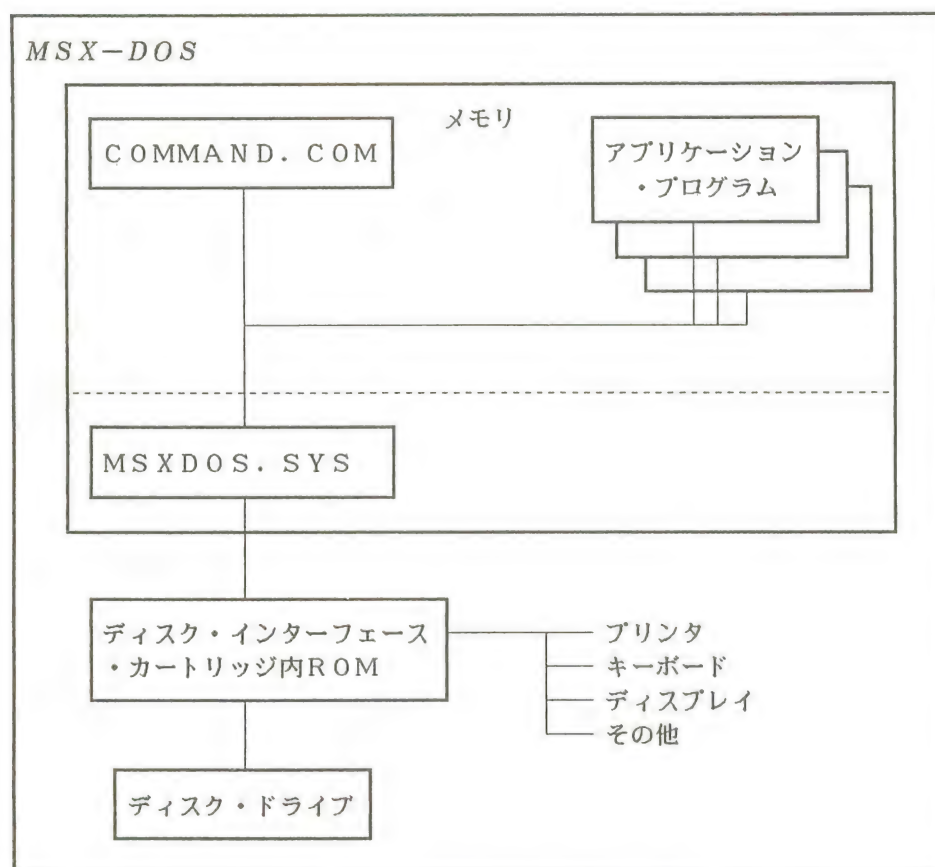
③COMMAND.COM

COMMAND.COMは、入力されたコマンドを受付けて、ファイル名の変更、削除といったファイルの管理やプログラムの実行を行います。

※MSXDOS.SYSとCOMMAND.COMは、システム・ディスクによって供給されます。

A.3 MSX-DOS概念図

MSX-DOSの概念図を図A.1に示します。この図は、“ディスク・インターフェイス・カートリッジ内ROM”、“MSXDOS.SYS”、“COMMAND.COM”および各周辺機器が、どのようにしてMSX-DOSを構成しているかを表しています。



図A.1 MSX-DOSの概念図

付録B

アプリケーション・プログラムの紹介

現在、MSX-DOSのもとでは次のようなアプリケーション・プログラムを利用することができます。

MSX-C	(株)アスキー
MSX-S BUG	(株)アスキー
α -FORTRAN	(株)ライフポート
α -PASCAL	(株)ライフポート
α -COBOL	(株)ライフポート
BDS-C	(株)ライフポート
ASM/FORTH	(株)インフォメーション・オフアリング・システム

付録C 関連書籍の紹介

現在、MSX-DOS関連の書籍としては、次のようなものが出版されています。また、MSX-DOSはMS-DOS、CP/Mと共通性をもっているので、この2つに関連する書籍を参考としても良いでしょう。

MSXテクニカル データブック1	アスキー	マイクロソフトFE本部
MSXテクニカル データブック2	アスキー	マイクロソフトFE本部
MSX2テクニカルハンドブック	アスキー	アスキー出版局
MSX早わかり事典	朝日新聞電子計算室編	朝日新聞社
入門MS-DOS	村瀬康治	アスキー出版局
応用MS-DOS	村瀬康治	アスキー出版局
実用MS-DOS	村瀬康治	アスキー出版局
入門CP/M	村瀬康治	アスキー出版局
実習CP/M	村瀬康治	アスキー出版局
応用CP/M	村瀬康治	アスキー出版局
標準MS-DOSハンドブック	村瀬康治	アスキー出版局

用語解説

“〜” コマンド

本書ではMSX-DOSに備えられているコマンドを、特に“〜”コマンドと書き表します。

アセンブラ

コンピュータに備えられたマイクロプロセッサの命令に対応するコード（インストラクションという）で書かれたソース・プログラムからコマンド・ファイルを作り出すプログラムをいいます。MSX-DOSで利用できるアセンブラの代表的なものとして、MSX-M80があります。

アプリケーション・プログラム

応用プログラムともいい、C、FORTRAN等のプログラミング言語、Multiplan等のビジネス・アプリケーションなどがあります。

外部記憶装置

コンピュータの外部に接続され、プログラムやデータを記憶し保存する目的で使用される装置をいいます。MSXで用いられる外部記憶装置には、データ・レコーダ（テープ・レコーダ）、クイック・ディスク、フロッピー・ディスク・ドライブなどがあります。

キーボード

コンピュータで、文字や数字を入力するために用いられる装置の一つで、多数の鍵を持つためこの名があります。

コンパイラ

プログラミング言語のうちソース・プログラムをコマンド・ファイルに変換する機能を備えたものをいいます。MSX-DOSで利用できるコンパイラの代表的なものとしてMSX-Cがあります。

ソース・プログラム

アセンブラやコンパイラがコマンド・ファイルを作り出す元となるプログラムをいいます。ソース・プログラムは、アセンブラではそのコンピュータのCPUの命令に対応するコマンドで、コンパイラではそのプログラミング言語のコマンドで書かれています。

ソフトウェア

コンピュータのプログラムを総称してソフトウェアといいます。

中央処理装置 →CPU

ディスク

ディスケット、フロッピー・ディスク、フレキシブル・ディスケットなどの呼称がありますが、本書ではディスクと書き表します。

ディスク・インターフェイス・カートリッジ

MSXコンピュータにドライブを接続するために用いられる装置で、カートリッジ・スロットに差し込んで使用します。

ディスプレイ

コンピュータが出力する文字や図形などの情報を表示する周辺機器をいいます。ビデオ・コンソール、ビデオ端末、CRT、モニタなどの呼称がありますが、本書ではディスプレイと書き表します。

ドライブ

フロッピー・ディスク・ドライブ、ディスク・ドライブ、FDDなどの呼称がありますが、本書ではドライブと書き表します。

内部記憶装置 →メモリ

ハードウェア

コンピュータ本体と周辺機器を総称してハードウェアといいます。
(cf. ソフトウェア)

プリンタ

コンピュータが出力する文字や図形などの情報を用紙に印刷する周辺機器をいいます。

プログラム

コンピュータに行わせる処理の手順を、そのCPUに備えられた命令で書き表したものをいいます。

プログラミング言語

プログラムを作るための言語で、BASIC、C、FORTRANなどがあります。

マイクロプロセッサ →CPU

メモリ

プログラムやデータを記憶し、処理を実行するためにコンピュータに備えられた装置で、読み書きが自由にできるRAM(Random Access Memory)と読み出し専用のROM(Read Only Memory)とがあります。

CP/M

Control Program for Microprocessorの略で、デジタル・リサーチ社が開発した8ビット・マイクロプロセッサ用のディスク・オペレーティング・システムをいいます。

CPU

Central Processing Unit の略で、中央処理装置、マイクロプロセッサなどの呼称があります。コンピュータの中心となる装置で、演算、データ処理などを行います。MSXコンピュータはCPUとしてザイログ社のZ80A相当品を使用しています。

MS-DOS

Microsoft Disk Operating System の略で、マイクロソフト社が開発した16ビット・マイクロプロセッサ(i8086系)用のディスク・オペレーティング・システムをいいます。

RAM →メモリ

ROM →メモリ

UNIX

ベル研究所が開発したマルチ・ユーザ/マルチ・タスク・オペレーティング・システムです。

索引

ア行	
アセンブラ	5 6
アプリケーション・プログラム	11
カ行	
外部記憶装置	1
外部コマンド	4
コマンド	1 2 3 4 9
コマンド行	2
コマンド・ファイル	5 6
コンバイラ	5 6
サ行	
システム・コール	3 5 7 6 9
システム・ディスク	9 10
ソース・プログラム	6
ソフトウェア	1 9 10
タ行	
中央処理装置	1
ディスク	1 2 3 7 7 8 9 10
ディスク・インターフェイス	
・カートリッジ	9 10
ディスク・オペレーティング・システム	1
ディスク・ファイル	1 3
ディスクROM	9
ディスプレイ	5 9 10
ディレクトリ	3 8
ドライブ	7
ナ行	
内部記憶装置	1
ハ行	
ハードウェア	9 10
バッチ・コマンド	4
ファイル	1 2 3 4 5 6 7 8 9
プリンタ	3 7 9 10
フロッピー・ディスク	1
フロッピー・ディスク・ドライブ	3 9
マ行	
マイクロプロセッサ	2 3
メモリ	1 10

ラ行	
リンカ	6
A	
autoexec. bat	2
C	
COMMAND. COM	9 10
CP/M	3 12
CPU	1
D	
del	4
Disk BASIC	3
dir	4
DOS	1 3
M	
MS-DOS	2 12
MSX	3 9 12
MSX-DOS	1 2 3 4 5 9 10 11
	12
MSX2	9
MSXDOS. SYS	9 10
O	
OS	1
R	
RAM	9
ren	4
T	
type	4
X	
XENIX	2

第2部

ユーザーズガイド





第1章	ユーザース・ガイドの使用法	
1. 1	ユーザース・ガイドの構成	1
1. 2	表記法	1
第2章	MSX-DOSの起動と終了	
2. 1	ディスク使用上の注意	3
2.1.1	MSX-DOSで使用するディスクの種類と名称	
2.1.2	ディスク取扱い上の注意	
2. 2	MSX-DOSの起動	4
2. 3	MSX-DOSの終了	6
2. 4	システム・ディスクの予備の作成	7
2. 5	ディスクに対する書き込み禁止	9
2. 6	ハードウェア構成による使用上の差異	9
2.6.1	時計機能	
2.6.2	MSXとMSX2	
2.6.3	仮想ドライブ機能	
2.6.4	複数のドライブを用いる場合のドライブ番号	
第3章	ファイルの概要	
3. 1	ファイルとは	16
3. 2	ディスク・ファイル	16
3.2.1	ファイル名	
3.2.2	ワイルドカード文字	
3.2.3	ドライブ指定	
3. 3	特殊なファイル名	20
第4章	コマンドの概要	
4. 1	コマンドとは	22
4.1.1	コマンドの概念	
4.1.2	コマンドの書式	
4. 2	コマンドの種類	22
4.2.1	内部コマンド	
4.2.2	外部コマンド	
4.2.3	バッチ・コマンド	

第5章 コマンド一覧	
BASIC	27
COPY	28
DATE	34
DEL (ERASE)	36
DIR	37
FORMAT	39
MODE	41
PAUSE	42
REM	43
REN (RENAME)	44
TIME	45
TYPE	47
VERIFY	48
第6章 特殊キーの機能	
6.1 特殊キーとは	49
6.2 コマンド行の編集機能	49
6.2.1 編集機能	
6.2.2 編集機能の使用例	
6.3 その他の機能	54
第7章 バッチ処理	
7.1 バッチ・ファイルの作成	55
7.2 パラメータの使用例	55
7.3 MSX-DOS起動時の自動実行機能	57
付録A MSX-DOSメッセージ一覧	
A.1 入力待ちメッセージ	59
A.2 エラー・メッセージ	64
※用語解説	69
※索引	71

第 1 章 ユーザーズ・ガイドの使用法

1. 1 ユーザーズ・ガイドの構成

第2部のユーザーズ・ガイドは、次のような構成になっています。

第1章では、ユーザーズ・ガイドの構成とコマンド等の説明に用いる表記法について解説しています。

第2章では、MSX-DOSを実際に動かす時の手順について解説しています。

第3章では、MSX-DOSでファイルを扱うために必要な知識について述べています。

第4章では、MSX-DOSのコマンドの概要について述べています。

第5章では、MSX-DOSの個々のコマンドについて解説しています。

第6章では、MSX-DOSの持つ有用な機能である特殊キーの使い方と、コマンド行の編集機能について解説しています。

第7章では、MSX-DOSの特徴であるバッチ処理について解説しています。

付録Aでは、MSX-DOSが表示する様々なメッセージとその対処法について述べています。困ったことがあった時は、この付録Aを参照してください。

1. 2 表記法

ユーザーズ・ガイドでは、コマンド等の解説に次のような表記法を用いています。

[] : 角型カッコ
コマンド、ファイル名などが角型カッコに囲まれている場合は、それが省略可能な項目であることを示します。

< > : 山型カッコ
山型カッコは、その中で指示されている項目を入力することを示します。
(例) copy [<ファイルスベック 1 >] [<ファイルスベック 2 >]
rem [<コメント >]

{ } : 大カッコ
大カッコに囲まれているものは、その中にあるもののうちの1つを必ず入力しなければならないことを示します。

| : 縦線
縦線で区切られた項目から1つを選んで入力します。

(例) VERIFY {ON|OFF}

... : 省略記号
直前の項目が必要に応じて何回も繰り返されることを示します。

CAPS : 大文字
大文字は、そのつづり通りに入力しなければならないコマンドまたはパラメータの一部分であることを示します。

— : 囲み
大文字がこの“囲み”内にある場合、キーボード上にあるその文字が書かれているキーを押すことを表します。また2つの囲みが“+”で結ばれている場合は、“+”の直前にある囲みのキーを押しながら“+”の直後のキーを押すことを示します。

(例) CTRL+C …… CTRLキーを押しながらCのキーを押す。

□ : 区切り記号
“□”はコマンドとパラメータ、あるいはパラメータとパラメータとの間の区切りを示します。区切り記号にはスペース、タブ、カンマが使用できます。

また、実際の使用例では以下のような記号を用います。

網掛け : 網掛けは、キーボードより入力しなければならない項目を示します。

□ : 本書では、区切り記号にすべて空白を用います。

⏏ : リターンキーを押すことを示します。

■ : ディスプレイに表示されるカーソルの位置を示します。

(例)

A>dir a:b ⏏

MSX DOS SYS 2432 85-08-23 9:29p

COMMAND COM 6656 85-09-02 10:10p

2 files 352256 bytes free

A>■

第2章 MSX-DOSの起動と終了

2.1 ディスク使用上の注意

2.1.1 MSX-DOSで使用するディスクの種類と名称

MSX-DOSでは、次の2つのタイプのディスクを使用します。

① 3.5インチ・マイクロ・フロッピー・ディスク 1DDタイプ

1DDとは片面倍密度倍トラック(Single sided, Double density, Double track)を示し、フォーマット時の記憶容量は 360Kバイトです。

② 3.5インチ・マイクロ・フロッピー・ディスク 2DDタイプ

2DDとは両面倍密度倍トラック(Double sided, Double density, Double track)を示し、フォーマット時の記憶容量は 720Kバイトです。

図2.1 に3.5インチ・マイクロ・フロッピー・ディスクの各部の名称を示します。

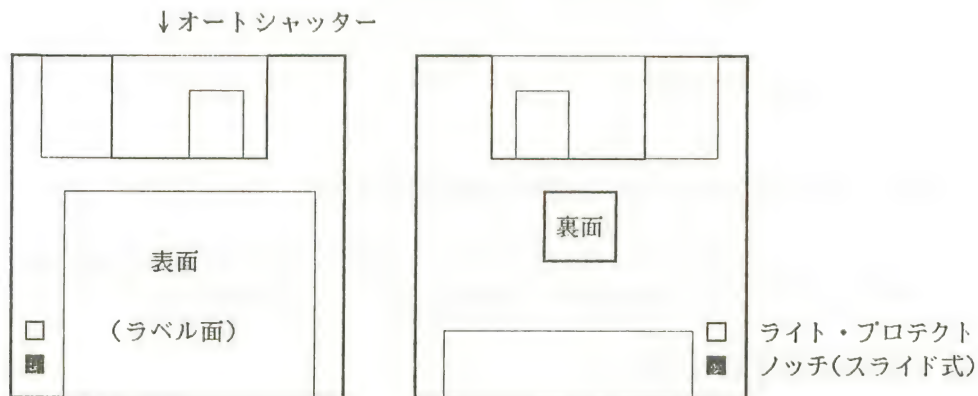


図2.1 3.5インチ・マイクロ・フロッピー・ディスクの各部の名称

ライト・プロテクト・ノッチについて

上の状態、つまりノッチ (■) が下にある状態では、書き込み禁止になっています。このとき、□の方は表と裏に貫通した穴となります。ノッチを上にはスライドさせると、□の穴がふさがって、書き込み可能となります。

2.1.2 ディスク取扱い上の注意

ディスクは薄い合成樹脂の円盤に磁性体を塗ったものなので、取扱いには細心の注意が必要です。3.5インチ・マイクロ・フロッピー・ディスクは、ハード・ケースに入っているため扱いが楽ですが、使用に際しては次のようなことに留意してください。

- ① ディスクのシャッターを開かないようにしてください。誤ってディスク面に触れると、汚れて読み書きができなくなります。
- ② 強い磁気をもっているもの、たとえばスピーカーなどのそばには、絶対に置かないようにしてください。書き込まれているデータが消えてしまいます。
- ③ 火気に近付けないでください。熱のためにディスク面が歪んで読み書きできなくなります。
- ④ 水などの液体を掛けないようにしてください。ディスクのみならずドライブの故障の原因となります。

この他、本来の目的を外れた使い方をしないようにしてください。

2.2 MSX-DOSの起動

MSX-DOSを起動するには、以下の手順で行います。

- ① MSXコンピュータとディスプレイ、ディスク・ドライブ、プリンタ（必要に応じて装備します）などの周辺機器が正しく接続されていることを確認します。
- ② 周辺機器の電源を入れます。
- ③ システム・ディスクをドライブAに挿入します。ここで“ドライブA”とは、次のようなドライブを指します。
 - ※接続しているドライブが1台の場合は、そのドライブ。
 - ※接続しているドライブが2台の場合は、ディスク・インターフェイス・カートリッジに接続されているドライブ。
 - ※なお本体内蔵型のドライブを使用している方は、「2.6.4 複数のドライブを用いる場合のドライブ番号」を参照してください。

④ MSXコンピュータの電源を入れます。

ここまででMSX-DOSがディスクからメモリに読み込まれ、ディスプレイに次のように表示されます。

```
MSXDOS.SYSのバージョン番号を表す。  
↓  
MSX-DOS version x.xx  
Copyright 1984 by Microsoft  
  
COMMAND version x.xx  
↑  
COMMAND.COMのバージョン番号を表す。
```

i. 時計機能を装備していないMSXコンピュータの場合、ディスプレイには続いて次のように表示されます。

```
Current date is Sun 84-01-01  
Enter new date: █  
    ↳これがカーソル
```

ここで日付を入力してください。入力を間違えた場合は、BS（バックスペース）キーを押してカーソルをその場所まで戻し、入力しなおします。

```
Enter new date: 85-9-2█
```

```
A> █
```

ii. 時計機能を装備したMSXコンピュータでは、内蔵した時計の時刻と日付が自動的に参照されるので、日付の入力を求めません。

```
MSX-DOS version x.xx  
Copyright 1984 by Microsoft
```

COMMAND version x.xx

A> 

MSX-DOSが起動すると、ディスプレイに“A>”という記号が表示されます。この記号はMSX-DOSの入力促進記号（プロンプト）で、これが表示されている時はコンピュータが入力を受け付ける状態であることを示しています。この状態を『MSX-DOSのコマンド入力待ち状態にある』といい、プロンプトに続くコマンド行に様々なコマンド名を入力し、プログラムを実行することができます。

プロンプトにある“A”は、現在コンピュータが優先して使用するドライブ（デフォルト・ドライブ）の名前で、“B>”ならばデフォルト・ドライブがドライブBであることを示します。ここで“d:”（dはドライブ名）を入力すると、デフォルト・ドライブを変更することができます。この結果プロンプトのドライブ名は指定したものに変わります。たとえばデフォルト・ドライブをAからBに変更する場合、次のように入力します。

A> B: 

B>

なお、MSX-DOSを起動した時点でのプロンプトは“A>”です。

2.3 MSX-DOSの終了

MSX-DOSでの処理が済んだら、次のような手順でMSX-DOSを終了します。

- ① 最後に実行したコマンドまたはプログラムの処理が終了していることを確認します。
処理が終わっていれば、ディスプレイにはプロンプトが表示されています。
- ② ディスク・ドライブからディスクを取り出します。
- ③ MSXコンピュータの電源を切ります。
- ④ 各周辺機器の電源を切ります。

最後に実行したコマンドまたはプログラムの処理が終了していないうちにコンピュータ

の電源を切ると、ディスクに記録されていたファイルが破壊されてしまう可能性があるの
で注意が必要です。

2. 4 システム・ディスクの予備の作成

MSX-DOSを初めて起動した時には、必ずシステム・ディスクの予備を作ります。
ディスクには、いつどんな事故が起こるかわかりません。大切なディスクが破損して読み
書きができなくなるとは大変です。予備のディスクを作り、マスター・ディスクは保管し
ておいて通常はこちらの方を使うようにしてください。予備のディスクは、“f o r m a
t”と“c o p y”の2つコマンドを使って、簡単に作ることができます。

予備のディスクの作成は次の手順で行います。

- ① マスター・ディスクのライト・プロテクト・ノッチを書き込み不可側に動かします。
- ② マスター・ディスクをドライブAに入れます。
- ③ 新しいディスクのライト・プロテクト・ノッチが書き込み可側になっていることを
確認してドライブBに入れます。
- ④ プロンプトよりf o r m a tと入力します。

```
A>format [F]           ↓ドライブBのディスクをフォーマット
Drive name? (A, B)  B
Strike a key when ready [X] ←何かキーを押すとこの
Format complete      カーソルは消える
                        ↖ フォーマットが終了すると表示される
A> [X]
```


なお2DDタイプのドライブを使用している場合は、次のようなメッセージが表示され
ます。これは2DDタイプの場合の一例なので、もしこれと違うメッセージが表示された
時は、そのディスク・ドライブのマニュアルを参照してください。

```
A>format [F]
Drive name? (A, B) [X]
```

```

Drive type
1...single sided, 8sectors
2...single sided, 9sectors
3...double sided, 8sectors
4...double sided, 9sectors

```

?  ←フォーマットの形式を選ぶ（標準はフォーマット Xはこの例では2）

```

Strike a key when ready 
Format complete

```

A> 

- ⑤ “copy” コマンドを起動し、システム・ディスクを新しいディスクにコピーします。

```




A>copy *.*.*b:  ←ドライブAにあるすべてのファイルを
MSXDOS      SYS      ドライブBにコピーするという意味。
COMMAND     COM
          2 files copied

```

A> 

- ⑥ これで予備のディスクができました。それでは実際に予備ができたかどうか “dir” コマンドで確認してください。

```

A>dir 
MSXDOS      SYS      2432  85-08-23   9:29p
COMMAND     COM      6656  85-09-02  10:10p
          2 files  352256 bytes free
A>dir *.*b: 
MSXDOS      SYS      2432  85-08-23   9:29p
COMMAND     COM      6656  85-09-02  10:10p
          2 files  352256 bytes free
A> 

```

- ⑦ 最後に予備のディスクでMSX-DOSを起動し、正しく予備が作れたかどうかを確認

認します。今ドライブBに入っているディスクをドライブAに挿入します。そして一度本体の電源を切り再び電源を入れるか、またはリセットします。

マスター・ディスクの時と同じようにMSX-DOSが起動すれば、予備のディスクは完全です。起動しなかった場合は、この手順でもう一度やり直してください。

2.5 ディスクに対する書き込み禁止

新しいディスクにコピーする時にディスクのライト・プロテクト・ノッチが書き込み不可側になっていると、次のメッセージが表示されます。

```
Write protect error writing drive B
Abort、 Retry、 Ignore?
```

このメッセージは、『ドライブBに入っているディスクが書き込み禁止になっています。処理を中止しますか、もう一度処理しますか、それともエラーを無視して処理を続けますか。』という意味です。MSX-DOSはディスクが書き込み禁止になっているためにコピーができず、どうすればよいかをたずねているのです。この場合は、ディスクを書き込み可にしなければコピーすることはできないので、まずコピーするディスクのライト・プロテクト・ノッチを書き込み可の側に動かします。それから“R”を押せば(RetryのRで、もう一度同じ処理をすることを示します)、コピーすることができます。もちろん“A”を押して(AbortのAで、その処理を中止することを示します)、始めからコピーをやり直してもかまいません。

エラー・メッセージの1行目はこの例の他にもいくつかあるので、付録Aを参照してください。2行目のメッセージに対しては“A”、“R”または“I”のいずれかのキーを押します。通常は“A”(中止)か“R”(再試行)を用い、“I”(無視)は他に方法がない場合しか使いません。

2.6 ハードウェア構成による使用上の差異

MSX-DOSは、時計機能の有無、MSXかそれともMSX2か、ディスク・ドライブが何台接続されているかなど、コンピュータを構成するハードウェアの違いによってその運用法が異なります。

2.6.1 時計機能

時計機能はMSXでは機種によっては装備されており、MSX2では標準装備になっています。

時計機能を装備したMSXコンピュータでは、日付、時刻を自動的に更新し、うるう年、月の大小、曜日なども管理されます。またMSX-DOSの起動時には、時計が持つ日付と時刻が自動的に参照されるので、日付の入力は必要ありません。ただし長時間電源を切ったままにしておくと時計の動作が停まり、正しい時刻と日付を示さなくなることがあります。そのような場合は、“date”コマンドと“time”コマンドを使って、正しい日付と時刻に修正してください。なお“date”コマンド、“time”コマンドについては第5章を参照してください。

2.6.2 MSXとMSX2

MSXとMSX2では、ディスプレイの最大表示文字幅が異なり、MSXでは40字、MSX2では80字になっています。なおMSX2の拡張機能として、文字の表示幅、画面の前景色、背景色、周辺色、ビープ音などを、起動時に自動的に設定することができます。これらのものは、Disk BASICで好みの値に設定し“set screen”命令でMSXコンピュータに記憶され、起動時にはこの値が用いられます。

2.6.3 仮想ドライブ機能

MSX-DOSは、1台のドライブでも運用が可能なDOSです。これはMSX-DOSが、1台のドライブをあたかも2台のドライブが存在するかのごとく動作させる、仮想ドライブ機能を備えているためです。

表2.1 に、内蔵型のディスクを持たず、ディスク・インターフェイス・カートリッジが1台の場合の、仮想ドライブ機能の有無を示します。

表2.1 仮想ドライブ機能の有無

ディスク・インターフェイス・カートリッジ	ドライブ数	仮想ドライブ機能
1 台	1 台	有
1 台	2 台	無

i. 仮想ドライブ機能が動作している場合の操作方法

MSX-DOSは、1台のディスク・インターフェイス・カートリッジに1台のドライブ、あるいは複数のディスク・インターフェイス・カートリッジに対してそれぞれ1台のドライブというシステム構成で起動した場合に、標準で仮想ドライブ機能が動作します。

MSX-DOSは2つの仮想ドライブを扱うコマンドを受け取ると、そのコマンドの処理中に次のようなメッセージを表示します。

```
Insert diskette for drive A:  
and strike a key when ready  
(ドライブA用のディスクを入れ、どれかキーを押してください)
```

ここで、その処理に必要なもう1枚のディスクと入れ換えます。するとMSX-DOSはそのディスクをもう1台の仮想ドライブとみなして処理を行います。その処理が終って、もとのディスクに対する処理が必要になると、再びディスプレイに次のようなメッセージを表示します。

```
Insert diskette for drive B:  
and strike a key when ready  
(ドライブB用のディスクを入れ、どれかキーを押してください)
```

このメッセージにしたがって処理を行えば、2台のドライブを必要とする処理を、1台のドライブで行うことができます。このように仮想ドライブ・シミュレータは、MSX-DOSを1台のドライブで運用する上で、重要な役割を果たしています。

ii. 仮想ドライブ機能を動作させない場合

2台のディスク・インターフェイス・カートリッジを用いて、それぞれに1台のドライブを接続した場合など、仮想ドライブ機能を用いるとこえて操作が複雑になることがあります。その場合には、起動時に音が鳴るまでCTRLキーを押し続けて、仮想ドライブ機能を動作させないようにしてください。

2.6.4 複数のドライブを用いる場合のドライブ番号

図2.2～2.7に、複数のドライブを接続する場合、どのようにドライブ番号が割り当てられるかを示します。カッコ内は仮想ドライブ機能が動作している場合です。起動後のドライブ番号がこの解説と異なる場合は、使用しているハードウェアのマニュアルを参照してください。

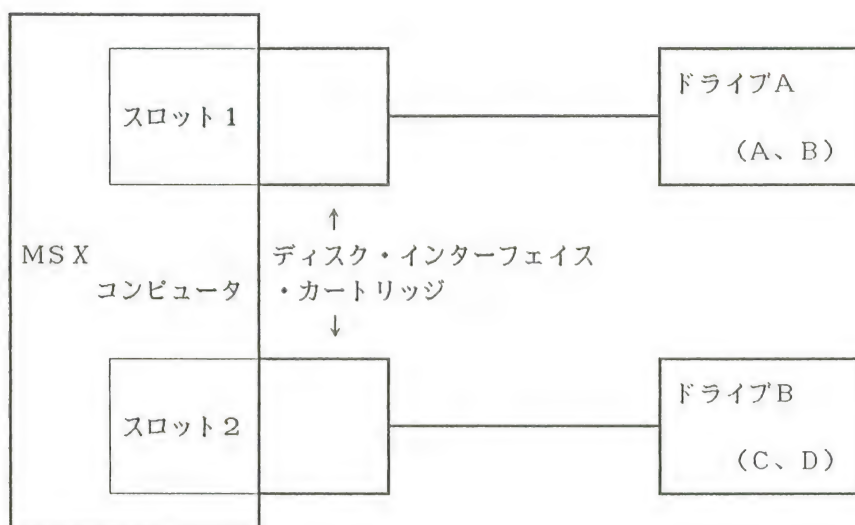


図 2.2 2台のディスク・インターフェイス・カートリッジにそれぞれ外付け型ドライブを1台ずつ接続した場合

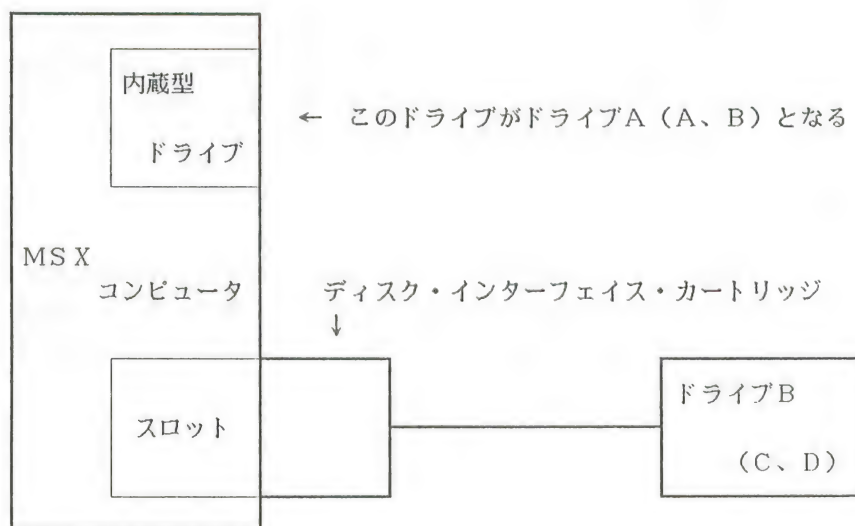


図 2.3 1台の内蔵型ドライブと、1台のディスク・インターフェイス・カートリッジに外付け型ドライブを1台接続した場合

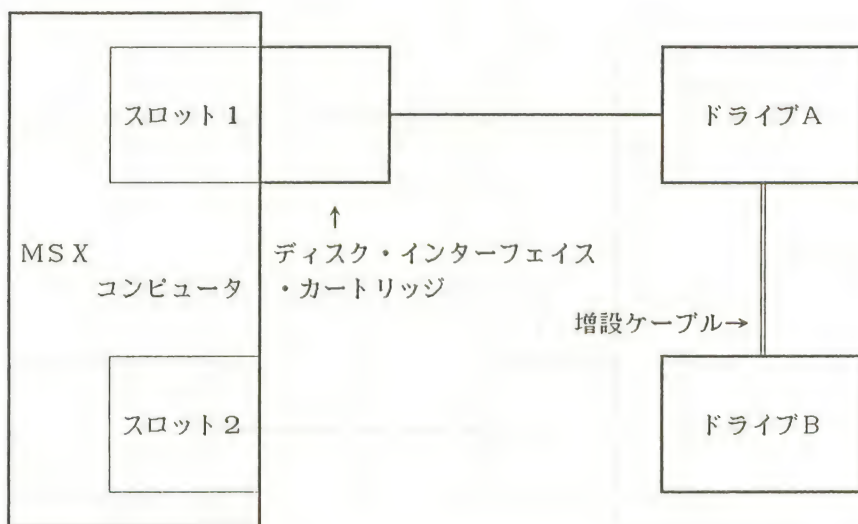


図 2.4 1 台のディスク・インターフェイス・カートリッジに
外付け型ドライブを 2 台接続した場合

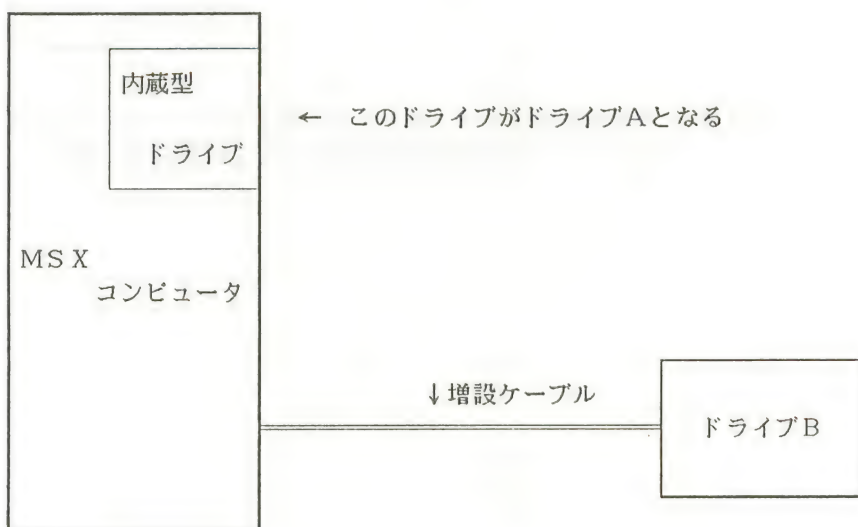


図 2.5 1 台の内蔵型ドライブと、増設ケーブルにより
外付け型ドライブを 1 台接続した場合

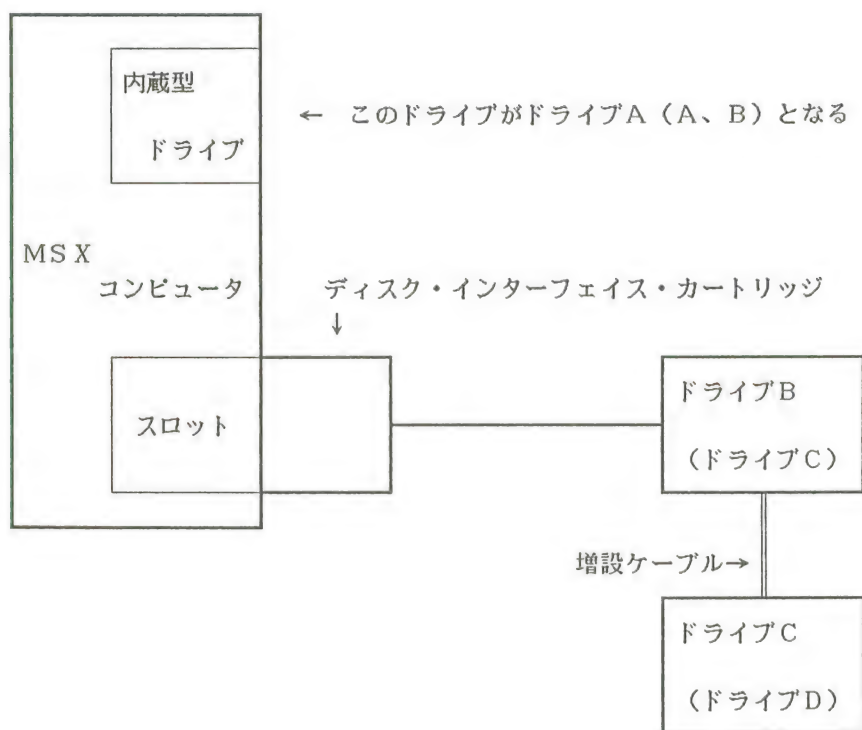


図 2.6 1 台の内蔵型ドライブと、1 台のディスク・インターフェイス・カートリッジに外付け型ドライブを 2 台接続した場合

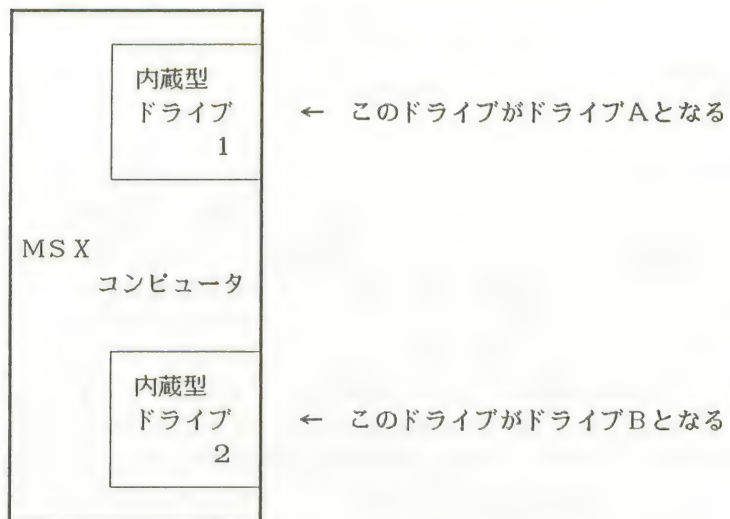


図 2.7 2台の内蔵型ドライブの場合

第3章 ファイルの概要

3.1 ファイルとは

コンピュータは、外部記憶装置としてフロッピー・ディスク・ドライブ、ウィック・ディスク、データ・レコーダ（テープ・レコーダ）や紙テープ装置などを使用します。これらの外部記憶装置に記録されたプログラムやデータを総称してファイルといいます。

DOSは、このようなファイルの登録、既にあるファイルへの追加、修正、ファイルの削除などの処理を行うプログラムを持っています。このようなプログラムを、ファイル・マネージメント・システム、あるいは単にファイル・システムと呼んでいます。

DOSが扱うファイルとは、普通ディスクに記録されるディスク・ファイルです。しかし、MSX-DOSでは、プリンタなどの周辺機器をもファイルとして管理しています。このため、周辺機器を扱うためのプログラムをわざわざ開発する必要がなく、周辺機器の取扱が容易になっています。各周辺機器には特殊なファイル名が割り当てられており、このファイル名はディスク・ファイル名と同様にコマンド行で使用することができます。

3.2 ディスク・ファイル

MSX-DOSではディスク・ファイルの管理を、それぞれのディスクに用意されているFAT(File Allocation Table)とディレクトリ(Directory)を用いて行っています。FATとは、ディスクでのファイルの配置状況を記録した地図であると考えることができます。また、ディレクトリは、ファイル名とそのファイルに関する情報を記録したメモであると考えればよいでしょう。DOSを利用するときはこのうちディレクトリにあるファイル名だけを意識していればよく、実際のファイル管理はすべてDOSが行います。

ここで第1部第3章で出てきた“dir”コマンドを思い起こしてください。“dir”コマンドはこのディレクトリの内容を表示するコマンドです。

```
A>dir
MSXDOS      SYS      2432  85-08-23    9:29p
COMMAND     COM      6656  85-09-02   10:10p
           2 files      352256 bytes free
A>
```

このように“dir”コマンドを用いてディレクトリを見ると、そのディスクにどんな

ファイルがあり、そのファイルの大きさといつ修正されたか、そしてディスクの残り容量がわかります。

3.2.1 ファイル名

ファイル管理の中心となるものは、そのファイルに付けられたファイル名です。ファイル名は、正確には1字以上8字以内のファイル名と3字以内の拡張子（ファイル・エクステンション）からなっていますが、拡張子は必ずしも必要なものではありません。またファイル名と拡張子の間はピリオド“.”で区切ります。ファイル名として使用できる文字は次のとおりです。

A～Z 0～9 \$ & # @ ! % ' ()
- { } ~ ひらがな カタカナ

なおアプリケーション・プログラムによっては一部の文字が使えないことがあります。その場合にはできるだけ A～Z、0～9 を使用してください。

アルファベットの小文字は、MSX-DOSによってすべて大文字に変換されます。これらの文字は、ファイル名や拡張子のどの部分でも使用することができます。

拡張子は、ファイルの種類を区別しやすくするために使用します。拡張子には、コンピュータでその使用が決っているものや、アプリケーションによっては習慣的に特定の拡張子を用いるものがあります。このような場合には、必ずその拡張子を使用してください。

① コンピュータでその使用が決っている拡張子

拡張子	ファイルの種類	例
.BAT	バッチ・ファイル	AUTOEXEC.BAT
.COM	コマンド・ファイル	COMMAND.COM
.SYS	システム・ファイル	MSXDOS.SYS

② 習慣的に使用する拡張子

拡張子	ファイルの種類
.ASM	アセンブラ・ソース・ファイル
.BAK	バックアップ・ファイル
.BAS	ベーシック・プログラム・ファイル
.C	C・ソース・ファイル
.COB	COBOL・ソース・ファイル
.CRF	クロスリファレンス・ファイル
.DAT	データ・ファイル
.FIF	FORTHソース・ファイル
.FOR	FORTRAN・ソース・ファイル
.HEX	インテル・hex・オブジェクト・ファイル
.LIB	アセンブラ・ライブラリ・ファイル
.M80	アセンブラ・ライブラリ・ファイル
.MAC	アセンブラ・ソース・ファイル
.PAS	PASCAL・ソース・ファイル
.PLI	PL/I・ソース・ファイル
.PRN	リステイング・ファイル
.REL	リロケータブル・オブジェクト・ファイル

なおアプリケーションによっては、使う拡張子が決っていてファイル名に拡張子を必要としないものもあります。

3.2.2 ワイルドカード文字

ワイルドカード文字とは、ファイル名を指定する際に、任意の1字あるいは文字列に対応して、その文字あるいは文字列の代りに用いることができる省略記号のことです。ワイルドカード文字を用いると、ファイルを指定する際にいくつかのファイルをまとめて指定することができます。ワイルドカード文字には、任意の1字に対応するクエスチョン・マーク(?)と、任意の文字列に対応するアスタリスク(*)の2種類があります。

“dir” コマンドを例にとって、ワイルドカード文字の使い方を解説します。“dir” コマンドにワイルドカード文字を用いれば、限定されたファイルについての情報を表示させることができます。

ドライブAに、次のようなディスクが入っていたとします。

```

A>dir
ABCDE      EXT      1664  85-07-17   9:18a
ABXDE      EXT      256   85-07-26   6:56p
ABYDE      EXT      1280  85-07-31   3:48p
ABCZDE     COM      9984  85-08-09  12:08p
XYZ        COM      8320  85-08-25   3:39a
           5 files      337920 bytes free

A>

```

i. クエスチョン・マーク (?)

“dir” のパラメータに “ab?de.ext” というファイル名を与えたとします。この場合 “dir” は、以下に示すようにファイル名が全体で5文字でABで始まりDEで終り、拡張子が “.EXT” であるすべてのファイルを表示します。

```

A>dir ab?de.ext
ABCDE      EXT      1664  85-07-17   9:18a
ABXDE      EXT      256   85-07-26   6:56p
ABYDE      EXT      1280  85-07-31   3:48p
           3 files      337920 bytes free

A>

```

なお “?” は、拡張子の中でも同様に使うことができます。

ii. アスタリスク (*)

ファイル名または拡張子の中に “*” があると、“*” はその位置以降の文字列を “?” に置き換えます。つまり、“*” は連続した “?” の略記と考えることができます。次の2つのコマンドは全く同じ意味で、そのディスクに記録されているすべてのファイルを表示します。

```

A>dir *.*.
A>dir ???????.????

```

次のコマンドは、拡張子が “.COM” である全てのファイルを表示します。

```

A>dir *.com ← dir ???????.com と同じ
ABCZDE     COM      9984  85-08-09  12:08p
XYZ        COM      8320  85-08-25   3:39a

```

```

2 files      337920 bytes free
A>

```

次のコマンドは、ファイル名がABCで始まる全てのファイルを表示します。

```

A>dir abc*. *  ← dir abc?????.??? 同じ
ABCDE      EXT      1664  85-07-17   9:18a
ABCZDE     COM      8320  85-08-25  12:08p
2 files      337920 files free
A>

```

3.2.3 ドライブ指定

MSX-DOSは、ファイルをディスク・ドライブごとに管理しています。そのため違うドライブに同じ名前のファイルがあっても、混同することなく扱うことができます。

MSX-DOSが管理するドライブの数は最大8台で、それぞれA、B、C……Hのアルファベットのドライブ名を持っています。ドライブを指定するときには、この文字の後ろにコロン（:）を付け“d:”というように指定しなければなりません。この文字とコロンの組み合わせを、“ドライブ指定”といいます。

また、ドライブ指定、ファイル名、拡張子をこの順で続けたものを、特に「ファイルスベック」といいます。ファイルスベックの書式は以下のとおりです。

[<ドライブ指定>] <ファイル名> [、<拡張子>]

例： A:BCDEFGHI.BAS

なおデフォルト・ドライブにあるファイルを指定するときは、ドライブ指定は省略することができます。

3.3 特殊なファイル名

MSX-DOSでは周辺機器に特殊なファイル名を割り当て、各周辺機器をディスク・ファイルと同じようにコマンド行で取り扱えるようにしています。この特殊なファイル名をデバイス・ファイル名といいます。デバイス・ファイル名には次のものがあります。

AUX : データ・レコーダなどの補助入出力装置との入出力や他のコンピュータとの通信のために用意してあります。起動時は、NULと同じ機能に設定して

あります。

CON : キーボードからの入力や、ディスプレイへ出力する場合に使用します。

NUL : コマンドが、その構文上入出力のファイル名を要求していても、特にファイルを作らない場合に使用します。

PRN : プリンタに出力する場合に使用します。LSTもこれと同じです。

デバイス・ファイル名を用いた例を以下に2つ上げます。

```
A>copy con:autoexec.bat
```

キーボードからの入力を“autoexec.bat”というファイル名でデフォルト・ドライブ上に作ります。

```
A>copy test.mac prn
```

“test.mac”というファイルの内容をプリンタに打ち出します。

なおデバイス・ファイルを使用する場合、たとえドライブ指定や拡張子が与えられていたとしても、デバイス・ファイル名が優先します。したがって、CON.LSTはファイルにはなりません。たとえば下のようなコマンドを入力してもCON.LSTというディスク・ファイルは作られず、SAMPLE.MACの内容がディスプレイに表示されます。

```
A>copy sample.mac con:lst
```

第4章 コマンドの概要

4.1 コマンドとは

4.1.1 コマンドの概念

BASICなどのプログラミング言語では、コマンドは1つのプログラムを構成する部分品でした。しかしMSX-DOSでいうコマンドとは、プログラミング言語のコマンドとは違い、プログラムそのものです。コマンドは、プロンプトに続くコマンド行にその名前を入力することで実行されます。

MSX-DOSで実際にコマンドの実行を受け持つのはシステム・コマンド・ファイルである“COMMAND.COM”です。すべてのコマンド（プログラム）は、“COMMAND.COM”によって解析され実行されます。

4.1.2 コマンドの書式

コマンドは、コマンド名とそれに続くパラメータからなっています。パラメータは、コマンドによって必要なものとそうでないものがあります。また、コマンド名とパラメータ、およびパラメータ相互の間は、スペース、タブまたはカンマによって区切られていなければなりません。コマンドの書式は以下のようになります。

A> [<ドライブ指定>] <コマンド名> [、<パラメータ>] . . .

これは、<ドライブ指定>で指定されたドライブにある<コマンド名>というコマンドを実行せよ、ということを示します。

4.2 コマンドの種類

コマンドには、コンピュータの内部にあらかじめ用意されている内部コマンドと、ディスクにファイルとして記録されていて、コマンド名が入力されるたびにメモリに読み込まれて実行される外部コマンド、MSX-DOSのバッチ処理機能により自動的に複数の内部コマンド、外部コマンドを実行するバッチ・コマンドの3種類があります。

4.2.1 内部コマンド

MSX-DOSには、以下の13種類の内部コマンドが備えられています。なお、それぞれのコマンドの詳細については、次章「コマンド一覧」を参照してください。

BASIC	DIR	REN	VERIFY
COPY	FORMAT	REM	
DATE	MODE	TIME	
DEL	PAUSE	TYPE	

4.2.2 外部コマンド

アセンブラやコンパイラなどを利用して作成した“.COM”という拡張子を持つファイルが外部コマンドです。システム・ディスクとともに提供される拡張子“.COM”のファイルもこれにあたります。

外部コマンドはファイル名をコマンド行から入力するだけで、ディスクから読み込まれて実行されます。なおコマンド・ファイルがデフォルト・ドライブにない場合は、コマンド名にドライブ指定を付けるか、デフォルト・ドライブをコマンド・ファイルがあるドライブに変更する必要があります。

以下に外部コマンドとデフォルト・ドライブの関係を示します。たとえばドライブA、Bに、それぞれ次のようなディスクが入っているとします。

```

A>dir
TEST          COM          384  85-07-25    6:16p
NEWDISK       BAT          78  85-08-03   10:02p
              2 files      360448 bytes free
A>dir b:
CAT           COM          8320 85-08-20    3:04p
              1 files      353280 bytes free
A>test
:
  [TESTというコマンド（プログラム）が実行される]
:
A>cat         ← “cat” を入力
Bad command or file name←CAT.COMがデフォルト・
A>b:cat       ドライブであるドライブA上に
:              なかったので、エラー・メッセ
  [CATが実行される]      ージが表示された。
:
A>b:         ← デフォルト・ドライブをドライブBに変更
B>cat        ← “cat” を入力
  
```

:
〔CATが実行される〕
:

B> 

4.2.3 バッチ・コマンド

バッチ・コマンドとは、拡張子が“.BAT”であるバッチ・ファイルに収められた一連のコマンドで、バッチ・ファイル名をコマンド行から入力すれば、収められているコマンドが次々と実行されます。バッチ・コマンドを利用することで、個々のコマンドをいちいち入力することなく、一連の処理を自動的に行うことができます。このためアセンブルやコンパイルなど、同じコマンドを使って何回も処理を行う場合には、非常に便利な機能です。



バッチ・ファイルがデフォルト・ドライブにない場合、バッチ・ファイル名の前にドライブ指定を付けるか、デフォルト・ドライブを変更しなければならないことは外部コマンドの場合と同様です。

前節でドライブAにあった“NEWDISK.BAT”が次のような内容であるとしします。ファイルの内容をディスプレイに表示させるには、“type”コマンドを使います。

```
A>type newdisk.bat
rem This is file NEWDISK.BAT
pause Insert new disk in drive b:
format
dir b:
```

A> 

このバッチ・ファイル名を入力すれば、バッチ・ファイルの内容が次々と実行されます。

```
A>newdisk
A>rem This is file NEWDISK.BAT
A>pause Insert disk in drive B:
Strike a key when ready  ←ドライブBに新しいディスク
                             を入れたらどれかキーを押す
A>format
Drive name? (A, B) b      ←ドライブBのディスクをフォーマット
Strike a key when ready  ←どれかキーを押す
Format complete          ←フォーマットが終了すると表示される
```


A>dir b:

File not found

←ドライブBのディスクにはファイルがない

A>■

第5章 コマンド一覧

本章では以下のコマンドについて解説します。コマンドはアルファベット順になっています。

BASIC	Disk BASICに移行します。
COPY	ファイルをコピーまたは連結します。
DATE	日付を表示、変更します。
DEL (ERASE)	ファイルを削除します。
DIR	ディレクトリを表示します。
FORMAT	ディスクを初期化します。
MODE	文字表示幅を設定します。
PAUSE	バッチ・コマンドの実行を一時停止します。
REM	バッチ・ファイルにコメントを書き込みます。
REN (RENAME)	ファイル名を変更します。
TIME	時刻を表示、変更します。
TYPE	ファイルの内容を表示します。
VERIFY	書き込みの際の検査の有無を設定します。

BASIC

内部コマンド

機能： Disk BASICを起動します。

書式： BASIC [＜ファイルスベック＞]

解説： MSX-DOSが起動されている状態からDisk BASICを起動したいときは、“basic” コマンドを用います。＜ファイルスベック＞によってDisk BASICのプログラム・ファイルを指定すると、Disk BASICを起動した後、そのプログラムを読み込んで実行します。

注： Disk BASICからMSX-DOSへ移行するには“_system”あるいは“call system”を用います。

例： Disk BASICを起動し、BASICのプログラムである“FUNCKEYS.INI”を実行します。このプログラムは、Disk BASICでファンクション・キーを設定して、再びMSX-DOSに戻るものです。

```
A>type funckeys.ini
100 KEY 1, "dir "
110 KEY 2, "ren "
120 KEY 3, "copy "
130 KEY 4, "del "
140 KEY 5, "type "
150 KEY 6, "time" +CHR$(13)
160 KEY 7, "date" +CHR$(13)
170 KEY 8, "format" +CHR$(13)
180 KEY 9, "verify "
190 KEY 10, "basic "
200 CLS:CALL SYSTEM
```

```
A>basic funckeys.ini
```

注意： MSX-DOSとDisk BASICではコンピュータのメモリ構成が異り、“basic” コマンドの実行によってメモリが切り替えられます。このためメモリを介してMSX-DOSとDisk BASICとの間でデータをやりとりすることはできません。

C O P Y

内部コマンド

機能1： ファイルを複写（コピー）します。

書式1： COPY [/A|/B] <ファイルスペック1> [/A|/B] <ファイルスペック2> [/A|/B]

機能2： ファイルを連結（付加）します。

書式2： COPY [/A|/B] <ファイルスペック1> [/A|/B] +<ファイルスペック2> [/A|/B] [. . . +<ファイルスペックn> [/A|/B]] <ファイルスペック> [/A|/B]

解 説： <ファイルスペック>の指定にはワイルドカード文字を使用することができます。普通“copy”コマンドは、ファイルを複写するために使用します。

i. 同じディスクにファイルを複写する

次の例は、“MSXDOS.SYS”というファイルを“MSXDOS.BAK”というファイル名で同じディスクに複写する場合のコマンド行を示しています。

```
A>copy msxdos.sys msxdos.bak
```

ここで<ファイルスペック1>と<ファイルスペック2>は、必ず違うファイル名でなければなりません。同じファイル名を指定した場合、“copy”コマンドは中断され（ファイルをそれ自身にコピーすることは許されません）、次のようなエラー・メッセージが表示されます。

```
A>copy msxdos.sys msxdos.sys
File cannot be copied onto itself
0 files copied
A>
```

ii. 違うディスクにファイルを複写する

この場合、<ファイルスペック2>には4つの形があります。

①省略された場合

コピーはデフォルト・ドライブに対して行われ、コピー後のファイルには<ファイルスペック1>と同じ名前が付けられます。次の例では、ドライブBの“msxdos.sys”がデフォルト・ドライブであるドライブAにコピー

されます。

```
A>copy┘b:msxdos.sys┘
```

<ファイルバック1>がデフォルト・ドライブ上のもので、<ファイルバック2>が省略されている場合、“copy” コマンドは中断され（ファイルをそれ自身にコピーすることは許されません）、次のようなエラー・メッセージが表示されます。

```
A>copy┘msxdos.sys┘  
File cannot be copied onto itself  
0 files copied  
A>┘
```

②ドライブ指定のみの場合

指定されたドライブに同じ名前でコピーします。

```
A>copy┘a:msxdos.sys┘b:┘
```

③ファイル名だけの場合

指定されたファイル名でデフォルト・ドライブにコピーします。

```
A>copy┘b:msxdos.sys┘msxdos.bak┘
```

④完全なファイルバックを指定した場合

指定されたドライブへ指定された名前でファイルをコピーします。

```
A>copy┘msxdos.sys┘b:msxdos.sys┘
```

copyコマンドの応用

i. “copy” コマンドのスイッチ

／Aスイッチと／Bスイッチはコピー・モードの指定に用います。“／A”はアスキー・モード (Ascii mode) の意味で、処理しているファイルの中にエンド・オブ・ファイル・マーク (EOF、ファイルの最後に付けられる) があると、そこでファイルが終わっているとみなして処理を打ち切ります。“／B”はバイナリ・モード (Binary mode) の意味で、EOFがあっても処理を続けます。

<ファイルスペック>にスイッチ<／A>または<／B>が付いている場合、そのスイッチは他のスイッチが見い出されるまで有効です。／Aが影響を持っている間に読み取られるファイルは、EOF以降が取り除かれます。

コピー先のファイル名に付加される／Aまたは／Bスイッチは、EOFがファイルの終りに置かれるかどうかを決定します。ファイルの書き込み中に／Aが影響を持つ場合、1つのEOFだけが付加されます。したがって、アスキー・ファイルを次のようなコマンドでコピーすると、もとのファイルのEOFは／Bスイッチのため除去されず、コピー先のファイルの／Aスイッチにより余分なEOFが付加されるので、結果として2つのEOFが付いたファイルができることになります。

```
A>copy a.b.mac/B cd.mac/A
```

次はファイルを連結するときにスイッチが付いた例で、実行ファイル“PROG.COM”に、アスキー・ファイルである定数データ“ERRS.TXT”を連結する場合を示しています。結果として必要なのがバイナリ・ファイルとすると終りのEOFは必要ないので、スイッチをこのように用います。

```
A>copy prog.com/a+errs.txt/b newprog.com/b
```

スイッチを指定しないで複写を行うと、バイナリ・モードでコピーされます。

```
A>copy/a a.b.txt cd.txt
```

この場合、“AB.TXT”の途中でEOFが含まれていると、その時点でコピーは打ち切られるので、“CD.TXT”の大きさは“AB.TXT”より小さくなります。なおこのとき“DE.TXT”には、EOFが最後の文字として付いています。

ii. ファイルの連結

“copy” コマンドを用いてファイルを連結することが可能です。連結は、“copy” コマンドのパラメータとして<ファイルスペック>を“+”でつなげながら並べるだけで行うことができます。

```
A>copy a b. txt + c d. txt + e f. txt g h. txt
```

この例では、“AB. TXT”、“CD. TXT”、そして“EF. TXT”を連結し、その結果を“GH. CRP”という名前でドライブAに書いています。

連結は、通常アスキー・ファイルに用いられます。バイナリ・ファイルを連結する場合、もしファイルの中にEOFが含まれていると、その時点でそのファイルのコピーを打ち切り、次のファイルを続けて連結コピーします。そのため、バイナリ・ファイルを連結する場合は/B スイッチによりバイナリ・モードに切り替えて行います。

```
A>copy /b a b. com + c d. com e f. com
```

この例では、“copy” コマンドは、“AB. COM”に“CD. COM”を連結し、“EF. COM”というファイルを作ります。

次の例はドライブBにある“TEST1. TXT”に、デフォルト・ドライブにある“TEST2. TXT”を付加する場合のコマンド行を示しています。

```
A>copy b: test1. txt + test2. txt b: test1. txt
```

ただし、もともになるファイルと付加するファイルが双方ともデフォルト・ドライブ上にある場合は、最後のパラメータを省略することができます。

iii. ワイルドカード文字の使用例

① 複写 (コピー)

ワイルドカード文字を利用すれば、ディスクのバックアップや特定のファイル名あるいは拡張子を持つファイルのみをコピーすることもできます。

次の例では、デフォルト・ドライブであるドライブAのすべてのファイルをドライブBにコピーします。

```
A>copy *,* b:
```

次の例では、デフォルト・ドライブにある拡張子が “.COM” であるファイルのすべてを、ドライブBにコピーします。

```
A>copy *.*.com b:
```

次の例では、ドライブBにあるファイル名が3字以下であるファイルのすべてを、デフォルト・ドライブにコピーします。

```
A>copy b:???.*
```

②連結（付加）

ワイルドカード文字を利用すれば、特定のファイル名あるいは拡張子を持つファイルすべてを連結して1つのファイルを作るなどの処理を行うことができます。

次の例では、拡張子が “.LST” であるすべてのファイルを連結して、“CONBIN.PRN” という名前のファイルを作ります。

```
A>copy *.*.lst conbin.prn
```

また、以下のようにいくつかのファイルを個々に連結したり、あるいは一つのファイルに連結したりすることもできます。たとえば、ドライブAに次のようなファイルがあるとして。

```
A>dir
FILE1      LST .....
ABCDE      LST .....
FILE1      REF .....
ABCDE      REF .....
           :
           :
```

このとき次のようなコマンドを入力すると、“FILE1.LST”と“FILE1.REF”が連結されて“FILE1.PRN”という名前のファイルが作られ、“ABCDE.LST”と“ABCDE.REF”が連結されて“ABCDE.PRN”という名前のファイルが作られます。

```
A>copy *.lst+*.ref *.prn
```

```
FILE1.LST
```

```
ABCDE.LST
```

```
2 files copied
```

```
A>dir
```

```
FILE1 LST .....
```

```
ABCDE LST .....
```

```
FILE1 REF .....
```

```
ABCDE REF .....
```

```
FILE1 PRN .....
```

```
ABCDE PRN .....
```

```
:  
:  
:
```

次の例では、“*.LST”に該当する全てのファイルを連結してから“*.REF”に該当する全てのファイルを連結し、さらにこの2つのファイルを連結して“COMBIN.PRN”というファイルを作ります。

```
A>copy *.lst+*.ref combin.prn
```

(注) 次のような場合、連結した結果作られるファイル名としてすでにディスクにあるファイルの名前を指定すると、そのファイルに連結するファイルの内容が重ね書きされ、結果としてもとのファイルの内容は書き換えられてしまうので、MSX-DOSは警告メッセージを表示します。

```
A>copy *.lst all.lst
```

```
Content of destination lost before copy
```

```
1 file copied
```

```
A>
```

次の例では、拡張子が“.LST”であるファイルのすべてを、“ALL.PRN”というファイルに付加しています。

```
A>copy all.prn+*.lst
```

DATE

内部コマンド

機能： 日付の表示、変更を行います。

書式： DATE [<年>-<月>-<日>]

解説： パラメータなしで入力した場合は次のように表示されます。

```
A>date
```

```
Current date is ww ww yy-mm-dd
```

```
Enter new date: 
```

ww、yy、mm、dd は内蔵の時計が示す日付をもとに表示されます。

ww : 曜日を示します。曜日は日付から自動的に計算されます。表示されるのは次のうちの一つです。

Sun Mon Tue Wed Thu Fri Sat

yy : 西暦を示し、下2桁が表示されます。

mm : 月を示し、2桁の数字です。

dd : 日を示し、2桁の数字です。

日付を変更する必要がある場合はここで を入力します。

パラメータとして日付を入力すると、そのまま新しい日付がセットされます。
なおこの場合は、メッセージは表示されません。

```
A>date 85-9-2
```

日付の入力には数字だけが使用できます（文字は許されません）。使用できるパラメータの範囲は次の通りです。

年 : 1980～2079

または80～99（1980～1999と解釈される）

または00～79（2000～2079と解釈される）

月 : 1～12（01～12）

日 : 1～31（01～31）

年月日の入力、ハイフン（-）かスラッシュ（/）で区切ってください。

日付は時刻とともにMSX-DOSによって管理されているので、時刻が24時になると自動的に更新されます。また月の大小やうるう年もチェックされ、正しく変更されます。ただし、時計機能がないMSXコンピュータの場合は更新はされませんので御注意下さい。

パラメータや区切り記号が誤っている場合は次のメッセージが表示されるので、もう一度正しく入力してください。

Invalid date ←日付の指定が違います
Enter new date:  ←日付を入力してください

注 意： 本体のROM BIOSが日本版以外のものでは、日付の表示と入力の形式が次のように変わります。

インターナショナル版 ： mm-dd-yyyy
ヨーロッパ版 ： dd-mm-yyyy

DEL (ERASE)

内部コマンド

機 能： <ファイルスペック>で指定されたファイルをすべて削除します。

書 式： DEL <ファイルスペック>
または
ERASE <ファイルスペック>

解 説： 次の例は、デフォルト・ドライブからファイル“TEST.MAC”を削除します。

```
A>del test.mac
```

“del”コマンドでは、削除するファイルの指定にワイルドカード文字が使用できます。たとえば拡張子が“.TXT”であるファイルをすべて削除するときは、次のように入力します。

```
A>del *.txt
```

ワイルドカード文字の詳細については、第3章「ファイルの概要」を参照してください。

ファイルの指定に“*.*”を用いると、ディスクにあるすべてのファイルを削除することになります。MSX-DOSは実行してもよいかどうかを確認するため、次のようなメッセージを表示します。全ファイルを削除してもよいときは、ここで“y”を入力します。

```
A>del *.*  
Are you sure (Y/N)?
```

なお“del”と同等の機能を持つコマンドとして“erase”が備えられています。

D I R

内部コマンド

機 能： ディスクに記録されているファイルについての情報を表示します。

書 式： D I R [<ファイルスベック>] [/ P] [/ W]

または

D I R [/ P] [/ W] [<ファイルスベック>]

解 説： “d i r” コマンドはディレクトリに記録されている、そのディスクにあるファイルの名前、大きさ、最後に修正された日付、時刻および表示したファイル数、ディスクの残り容量などの情報をディスプレイに表示します。

“d i r” コマンドでは“/ P”と“/ W”の2つのスイッチが使用できます。
/ Pスイッチは、ページ・モードを意味し、ディスプレイいっぱいに表示されたところで表示を中断します。表示を再開するには、任意のキーを押します。

/ Wスイッチはワイド・ディスプレイ・モードを意味し、ファイル名のみを1行に表示できるだけ表示します。1行の表示幅は“m o d e”コマンドで設定した値によります。

“d i r” コマンドにパラメータがない場合、デフォルト・ドライブ上にあるディスクのディレクトリの内容を表示します。デフォルト・ドライブAにあるディスクのディレクトリの内容を表示させるには、次のように入力します。

```
A>d i r
```

パラメータとしてドライブ指定のみがある場合、指定されたドライブにあるディスクのディレクトリの内容を表示します。ドライブBにあるディスクのディレクトリの内容を表示させるには、次のように入力します。

```
A>d i r b :
```

パラメータとして拡張子なしのファイル名だけがあるとき（ドライブ指定はあってもよい）、そのドライブ上にあるディスクのディレクトリを検索して、指定されたファイル名を持つすべてのファイルの情報を表示します。次の例では、ドライブBにあり、ファイル名に“T E S T 1”を持つすべてのファイルの情報を表示します。

```
A>d i r b : t e s t 1
T E S T 1      A S M      .....
T E S T 1      R E L      .....
```



```

TEST1      COM      .....
          3  files .....
A>

```

完全なくファイルスバック>を指定した場合、指定されたドライブにあるディスクのディレクトリからそのファイルを検索し、その情報を表示します。

```

A>dir \b test1.asm
TEST1      ASM      .....
          1  file   .....
A>

```

ファイル名のパラメータとしてワイルドカード文字を使用することができます。ワイルドカード文字については、第3章「ファイルの概要」を参照してください。なお、“dir”コマンドの呼び出しで同等の動作をするものを以下に上げます。

コマンド	同等のコマンド形式
dir	dir *.*
dir file	dir file.*
dir .ext	dir *.ext
dir .	dir *.

注 意： 表示幅が36桁未満の場合は、情報の一部が表示されないことがあります。このようなときは“mode”コマンドを用いて、表示幅を36桁以上に設定してください。

FORMAT

内部コマンド

機能： 新しいディスクを使用できるように初期化します。

書式： FORMAT

解説： “format” コマンドの処理はメッセージに従って対話方式で進められます。標準的なメッセージは以下の通りです。

i. ドライブが1DDタイプの場合

```
A>format
Drive name? (A, B)
Strike a key when ready
Format complete
```

↑
どれかキーを押す

ii. ドライブが2DDタイプの場合

```
A>format
Drive name? (A, B)
Drive type
1... 8sectors, single sided
2... 9sectors, single sided
3... 8sectors, double sided
4... 9sectors, double sided
?
Strike a key when ready
Format complete
```

↑
どれかキーを押す

“format” コマンドを中断するときは、CTRL+Cを入力します。このときディスプレイには次のようなメッセージが表示されます。

Aborted ←中止しました

注 : MSX-DOSで利用できるディスクの様式を表5.1 に上げます。

表 5.1 MSX-DOSで利用できるディスクの様式

ディスクの様式	1 D D 9sectors	2 D D 9sectors	1 D D 8sectors	2 D D 8sectors
記録可能な総ファイル数	112	112	112	112
F A T が占めるセクタ数	2	3	1	2
1トラックのセクタ数	9	9	8	8
サイド数	1	2	1	2
サイドあたりのトラック数	80	80	80	80
1セクタのバイト数	512	512	512	512
使用可能なセクタ数	708	1426	630	1268
使用可能な総バイト数	362496	730112	322560	649216

物理セクタ番号 1トラック8セクタの場合 1～8
 1トラック9セクタの場合 1～9
 トラック番号 0～79
 サイド番号 1 D Dの場合 0
 2 D Dの場合 0、1

3.5インチ型のMS X用ディスクドライブ装置では、全ての機種で片面80トラック（1 D D）フォーマットを読み書きできるようになっています。他の装置で使う可能性のあるディスクをフォーマットする場合は、このフォーマットを選ぶことが望ましいでしょう。

MODE

内部コマンド

=====

機 能： ディスプレイの表示文字幅を設定します。

書 式： MODE <表示文字幅>

解 説： 設定できる表示文字幅は、MSXでは1～40で41以上の値を設定するとエラーになります。MSX2の場合は1～80です。設定した値が1～32の場合はスクリーン・モード1（BASICのscreen 1）に、33～40（MSX2では33～80）の場合はスクリーン・モード0（BASICのscreen 0）に設定されます。

“mode” コマンドを実行するとディスプレイは消去されます。


PAUSE

内部コマンド

機 能： バッチ・コマンドの実行を一時停止します。


書 式： PAUSE [＜コメント＞]

解 説： バッチ・コマンドの実行を一時停止させる目的で使います。バッチ・ファイルにこのコマンドを書き込んでおくと、バッチ・コマンドの実行が“pause”のところで一時停止します。このときディスプレイには以下のようなメッセージが表示されます。

```
Strike any key when ready   
(準備ができたらどれかキーを押してください)
```



バッチ・コマンドの実行は、CTRL+C以外の任意のキーを押すと再開されるので、その間に必要な処理を行います。

CTRL+Cをタイプすると、続いて次のようなメッセージが表示されます。

```
Terminate batch file (Y/N)?   
(バッチ処理を中止しますか?)
```

ここで“Y”を入力するとバッチ・コマンドの実行は中止され、*MSX-DOS*のコマンド入力待ちに戻ります。“N”を入力した場合はバッチ・コマンドの実行が再開されます。

“pause”コマンドのパラメータにコメントを与えると、それをディスプレイに表示させることができます。たとえば下の例のように、使用者に指示を与えることも可能になります。

```
A>newdisk  
A>rem This is NEWDISK. BAT  
A>pause Insert disk in drive B:  
Strike a key when ready  
A>format  
Drive name (A,B) ? b  
Strike a key when ready  
Format complete  
  
A>
```

R E M

内部コマンド

=====

機 能： 何もしません。ただし結果として、“r e m”のあとに続くコメントをディスプレイに表示します。

書 式： R E M_ [<コメント>]

解 説： “r e m” コマンドは上記の動作以外に、他に何の影響も与えません。“p a
u s e” コマンドの例を参照してください。

REN (RENAME)

内部コマンド

機 能： 第1パラメータで指定したファイル名1を、第2パラメータのファイル名2に変更します。

書 式： REN <ファイル名1> <ファイル名2>

または

RENAME <ファイル名1> <ファイル名2>

解 説： デフォルト・ドライブ以外のファイル名を変更する場合は、ファイル名1を指定する際にドライブ指定が必要です。“ren” コマンドはファイル名の変更を行うものなので、ファイル名2でドライブを指定しても意味を持ちません。また“ren” コマンドで他のディスクにファイルを移動させることはできません。

“ren” コマンドではワイルドカード文字を使用することができます。このとき、ファイル名1で指定したファイルとファイル名2で指定したファイルとの間で、各文字が1対1に対応して処理されます。たとえば次の例では、“.LST” 拡張子を持つ全てのファイルの拡張子を“.PRN” に変えます。

```
A>ren   *.lst   *.prn
```

次の例では、ドライブB上のファイルABCDEをADCB Eという名前に変更することになります。

```
A>ren   b:abcde   ?d?b?
```

なお、“rename” は“ren” と同等のコマンドです。

TIME

内部コマンド

機能：時刻の表示、変更を行います。

書式：TIME [_ <時> [: <分> [: <秒>]]] {a | p}

解説：パラメータなしで入力した場合は次のように表示されます。

```
A>time
Current time is hh-mm-ss.tt {a | p}
Enter new time:
```

hh、mm、ss、tt、aまたはpは内蔵の時計が示す時刻です。

hh : 1～12の数字で時を示します。

mm : 0～59の数字で分を示します。

ss、tt : ssは0～59の数字で秒を示します。またttは $\frac{1}{100}$ 秒を示しますが表示される数字は“00”です。

aまたはp : 午前か午後かを示します。

表示された時刻を変更する必要がある場合はここでを入力します。

パラメータとして時刻を入力すると、メッセージは表示されずにそのまま時刻の変更が行われます。

時刻の入力には数字だけが使用でき、“.”は入力する必要がありません。また、最後の数字の後に“a”または“p”を付けて、午前(am)と午後(pm)を指定することができますが、これをつけないと24時間制で時刻を指定したとみなされます。従って次の2つの例では、最初のものは午前10時10分30秒に変更し、次のものは時刻を午後10時10分30秒に変更します。

```
A>time_10:10:30
A>time_10:10:30p
```

時刻の入力に使用できるパラメータの範囲は次の通りです。

時 : 0～23 (24時間制で入力したと見なされる)
または1～12 {a | p} (aまたはpを省略すると午前と見なさ

れる)
分 : 0~59
秒 : 0~59

時：分：秒の入力は、コロン（:）で区切って行います。パラメータや区切り記号が誤っている場合、次のメッセージが表示されるので、もう一度正しく入力してください。

Invalid time ←時刻の指定が違います
Enter new time:  ←時刻を入力してください

T Y P E

内部コマンド

機 能： <ファイルスベック>で指定したファイルの内容をディスプレイに表示します。

書 式： T Y P E ␣ <ファイルスベック>

解 説： 表示させるファイルの名前を捜すには“d i r”コマンドを使います。<ファイルスベック>の指定にはワイルドカード文字を使用できますが、ディレクトリを捜して最初に指定に該当したファイルの内容のみを表示します。

“t y p e”コマンドはアスキー・ファイルの内容を表示させるために用います。バイナリ・ファイルの内容を表示させると、ベル・コード、フォーム・フィード・コード、およびエスケープ・シーケンスを含むコントロール・シーケンスがディスプレイに送られ、正しい表示が行われません。

V E R I F Y

内部コマンド

機 能： ディスクに書き込みを行う際に、検査を行うかどうかを設定します。

書 式： V E R I F Y { O N | O F F }

解 説： “v e r i f y” コマンドは、書き込みに際してファイルが正しく書き込まれたことを検査するかどうかを設定します。“v e r i f y”を“o n”にすると、ディスクに書き込むごとに検査を行い、o f fにすると検査は行いません。

“v e r i f y o n”の場合は書き込みに時間がかかるので、通常は“o f f”にします。MSX-DOSの起動時には、“v e r i f y o f f”に設定されています。重要なファイルをコピーする場合、たとえばマスター・ディスクの予備を作るときなどには、“v e r i f y o n”にすると良いでしょう。

なお、ベリファイ機能はオプションであり、ドライブによっては行わないものもあります。

第6章 特殊キーの機能

6.1 特殊キーとは

MSX-DOSには、普通の文字キーの他に、特別な機能を持ついくつかの特殊キーがあり、これらのキーを使うことでMSX-DOSの操作がより便利になります。

この特殊キーには、CTRL、DEL、HOME、INS... などがあります。特殊キーのほとんどは単独で使用されますが、CTRLだけは、それを押しながら他の文字キーを押して使用します。なお以下の解説では、CTRL+文字は、CTRLキーを押しながら文字のキーを押すことを表します。

6.2 コマンド行の編集機能

6.2.1 編集機能

MSX-DOSには、直前に入力されたコマンド行を記憶し、それを再び実行したり、修正して実行したりする機能があります。この機能を利用すれば、同じようなコマンドを繰り返し入力する手間を減らすことができます。この記憶されたコマンド行をテンプレートといいます。

図6.1 は入力されたコマンド行がどのように記憶され、呼び出されるかを表したものです。

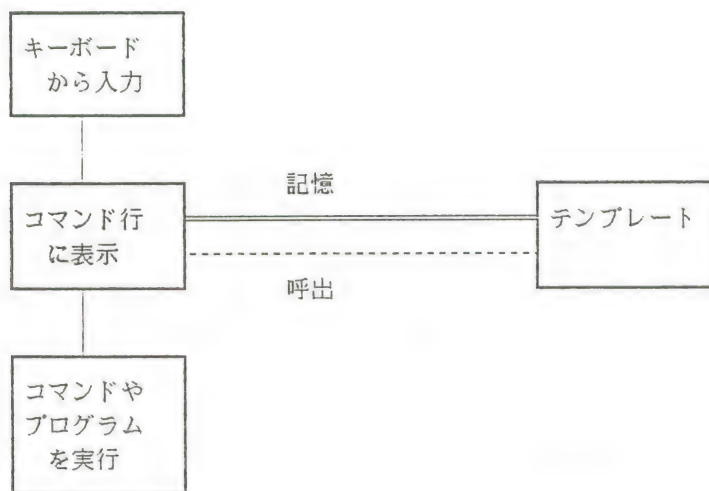


図 6.1 コマンド行とテンプレート

コマンド行の編集に使われる特殊キーの機能を表6.1 に示します。1つの機能に複数のキーが割り当てられている場合は、どのキーを用いても同じ効果があります。

表 6.1 特殊キーの機能

機 能	特殊キー	内 容
1 字コピー	<u>→</u> <u>CTRL+¥</u>	テンプレートからコマンド行へ 1 文字コピーします。
指定コピー	<u>SELECT</u> <u>CTRL+X</u>	テンプレートからコマンド行へ、指定された文字の直前までの文字をコピーします。
1 行コピー	<u>↓</u> <u>CTRL+</u>	テンプレート内に残っている全ての文字をコマンド行へコピーします。
1 字スキップ	<u>DEL</u>	テンプレート内の 1 文字を飛び越します（コピーしません）。
指定スキップ	<u>CLS</u> <u>CTRL+L</u>	テンプレート内の文字を指定された文字の直前まで飛び越します（コピーしません）。
取消	<u>↑</u> <u>ESC</u> <u>CTRL+^</u> <u>CTRL+[</u> <u>CTRL+U</u>	現在のコマンド行を取り消します。テンプレートの内容は変更されません。
挿入	<u>INS</u> <u>CTRL+R</u>	挿入モードに入ったり抜け出たりします。
後退	<u>←</u> <u>BS</u> <u>CTRL+H</u> <u>CTRL+]</u>	コマンド行から最後の 1 文字を除去します。
新しい 行を作る	<u>HOME</u> <u>CTRL+K</u>	コマンド行をテンプレートにコピーします（新しいテンプレートを作ります）。
行換え	<u>CTRL+J</u>	ディスプレイの表示のみ改行します。

6.2.2 編集機能の使用例

ここで実際に例を上げてコマンド行の編集機能を解説します。なお、テンプレートに関するキー操作は \leftarrow と同様にディスプレイには表示されません。ここではわかりやすくするために《 》で囲んで示します。

```
A>dir prog.com $\leftarrow$ 
```

このコマンド行は、ファイル“PROG.COM”の情報を表示するものです。このコマンド行を入力すると“dir”コマンドの実行とともに、コマンド行がテンプレートに記憶されます。もう一度このコマンド行を繰り返すには、 \downarrow と \leftarrow を使います。 \downarrow を押すと、まずテンプレートがコマンド行にコピーされ、次に \leftarrow を押すとコマンドとして実行されます。

```
A>《 $\downarrow$ 》dir prog.com $\leftarrow$ 
```

次に、ファイル“PROG.MAC”の内容を表示するとします。現在テンプレートには“dir prog.com”と記憶されているので、まずSELECTを押し、続いてCと押すことで“c”の直前までがコマンド行にコピーされます。

```
A>《SELECT-C》dir prog. $\leftarrow$ 
```

これに続けて“mac”と押すと、目的のコマンド行ができあがります。

```
A>dir prog.mac
```

ここで \leftarrow を押すと、このコマンド行が実行され、今度は“dir prog.mac”がテンプレートに記憶されます。では“type prog.mac”というコマンド行を作ってみます。それには次のようにキーを押します。

```
A>type INS $\leftarrow$ 《 $\downarrow$ 》
```

新しく入力された文字は直接コマンド行に入力されます。“dir”とそれに続くスペースの4文字が“type”に変換され、そしてコマンドとパラメータの間に必要なスペースを入れるためにINSとスペースを入力します。そして最後に、テンプレートの残りの“prog.mac”をコマンド行にコピーします。するとコマンド行は次のようになります。

```
A>type prog.mac $\leftarrow$ 
```

次にコマンド行の編集機能を用いてテンプレートを修正する手順を示します。ここでは“type”とすべきところを“byte”と入力してしまったとします。これを修正するにはHOMEを使います。HOMEは現在のコマンド行をテンプレートにコピーし、新しいテンプレートを作ります。HOMEを押すと新しいテンプレートを作った目印として、コマンド行の末尾に“@”が表示され、カーソルは次の行に移ります。

```
A>byte prog.mac《HOME》@
```

ここで、→と↓を使いテンプレートの内容を修正しながらテンプレートをコマンド行にコピーします。→はテンプレートから1文字をコマンド行にコピーする機能を持っています。間違えた“b”を“t”に、さらに“t”を“p”に置き換え、残りを↓でコピーします。

```
A>byte prog.mac@
t《→》yp《↓》e prog.mac
```

また同じことは、以下のようにDELおよびINSを用いても可能です。

```
A>byte prog.mac@
《DEL》《DEL》《→》t《INS》yp《↓》 prog.mac
```

これを分解すると次のようになります。

タイプ	:	コマンド行	
《DEL》	:		: 1文字目をスキップ
《DEL》	:		: 2文字目をスキップ
《→》	:	t	: 3文字目をコピー
《INS》yp	:	typ	: ypを挿入
《↓》	:	type prog.mac	: 残りをコピー

6.3 その他の機能

MSX-DOSにはコマンド行の編集に用いる特殊キーの他に、表6.2のような特殊キーがあります。

表 6.2 その他の特殊キー

機 能	特殊キー	内 容
中 止	<u>C T R L + C</u>	現在動作中のコマンドやプログラムの実行を中止します。
印 刷	<u>C T R L + P</u>	ディスプレイへの表示をプリンタへも出力します。
印刷取消	<u>C T R L + N</u>	“ <u>C T R L + P</u> ”の動作を取消します。
表示停止	<u>C T R L + S</u>	ディスプレイへの表示を一時停止します。任意のキーを押すと表示は再開します。

第7章 バッチ処理

7.1 バッチ・ファイルの作成

バッチ・コマンドを利用するには、まずディスク上にバッチ・ファイルを作成しなければなりません。バッチ・ファイルは、“copy”コマンドを用いて次のような手順で作成します。エディタを持っている方は、それを用いて作成することもできます。

キーボードからの入力を“AUTOEXEC.
BAT”というファイル名でドライブAのデ
ィスクにコピーする

```
A>copy con autoexec.bat
rem This is AUTO EXECUTE FILE
basic funckeys.ini
^Z      ←ファイルの終りには、必ずCTRL+Zを入力する
        1 file copied
A>
```

“^Z”はファイルがそこで終ることを示す記号で、エンド・オブ・ファイル・マークあるいは単にEOFといい、ファイルの最後には必ず書き込まれる必要があります。

上の例の“AUTOEXEC. BAT”がディスク上に作成されたかどうか“type”コマンドで確認すると以下ようになります。

```
A>type autoexec.bat
rem This is AUTO EXECUTE FILE
basic funckeys.ini
A>
```

7.2 パラメータの使用例

バッチ・ファイルでは、%のあとに1桁の数字を用いた仮パラメータを使用することができます。この仮パラメータは、バッチ・コマンド実行時にコマンド行に書かれたファイルスベックやスイッチ、パラメータと置き換えて処理されます。これを利用すれば、1つのバッチ・ファイルで様々なファイルに対する処理を行うことが可能です。

仮パラメータは、%0から%9までの10個が使用できます。%0はコマンド名自身と

置き換えられ、%1から%9までの9個が実パラメータと置き換えられます。次の例は仮パラメータを使用したバッチ・ファイルで、アセンブラを用いてコマンド・ファイルを作っています。

```
A>type asm.bat
rem This is ASM. BAT
rem START BATCH FILE
dir
m80 =%1. asm
m80 =%2. asm
l80 %1,%2,%1/n/e
dir
```

```
A>
```

これを実行すると……

test1.asmとtest2.asmをアセンブルして、コマンド・ファイルを作成する

```
A>asm test1 test2
A>rem This is ASM. BAT
A>rem START BATCH FILE
A>dir
MSXDOS      SYS      2432  85-08-23   9:29p
COMMAND     COM      6656  85-09-02  10:10p
M80          COM     20224  85-04-01  10:27p
L80          COM     10725  85-04-01  10:25p
ASM          BAT       117  85-07-17  12:24p
TEST1       ASM       256  85-07-18  12:20a
TEST2       ASM       256  85-07-18  12:23a
              7 files      317440 bytes free
```

```
A>m80 =test1. asm
```

```
No Fatal error(s)
```

```
A>m80 =test2. asm
```

No Fatal error(s)

A>180 test1, test2, test1/n/e

MSX.L-80 1.00 01-Apr-85 (C) 1981, 1985 Microsoft

Data 0100 0141 < 65>

43504 Bytes Free
[0000 0141 1]

A>dir
MSXDOS SYS 2432 85-08-23 9:29p
COMMAND COM 6656 85-09-02 10:10p
M80 COM 20224 85-04-01 10:27p
L80 COM 10725 85-04-01 10:25p
ASM BAT 117 85-07-17 11:24p
TEST1 ASM 256 85-07-18 12:20a
TEST2 ASM 256 85-07-18 12:23a
TEST1 REL 128 85-07-18 12:24a
TEST2 REL 128 85-07-18 12:25a
TEST1 COM 128 85-07-18 12:25a
10 files 314368 bytes free

A>

A> 

このように、%1、%2という2つの仮パラメータが、コマンド行から入力した順に test1、test2と置き換って処理されます。

7.3 MSX-DOS起動時の自動実行機能

“AUTOEXEC.BAT”というバッチ・ファイル名は、MSX-DOSに予約されています。“AUTOEXEC”とは自動実行(Auto execution)という意味で、MSX-DOS起動時に自動的に実行されるバッチ・ファイルです。

MSX-DOSが起動すると、コンピュータはドライブAに入っているディスクから“ AUTOEXEC.BAT”というバッチ・ファイルを捜し、見つければバッチ・コマンドとして実行します。“AUTOEXEC.BAT”がない場合は通常どおりに起動しま

す。

なお時計機能を持たないMSXコンピュータの場合、“AUTOEXEC.BAT”が実行されると、起動時に必要な日付を入力する手続は省略されますがMSX-DOSの動作に影響はありません。

付録A MSX-DOSメッセージ一覧

MSX-DOSは、動作中にいろいろなメッセージを表示します。このメッセージには、使用者に対してなんらかの処置を求める入力待ちメッセージと、エラーが起こったことを伝えるエラー・メッセージとがあります。

解説はメッセージのアルファベット順に、次のような構成で行います。なお、ディスプレイには対処法が表示されないので、そのメッセージに対する対処についても解説を加えます。

《表示されるメッセージ》

※《そのメッセージが表示される可能性があるコマンドなど》

意味：《メッセージの意味》

内容：《メッセージが表示された原因、内容など》

対処：《そのメッセージへの対処法》

なお“※”の項に“COMMAND”とあるものは、そのメッセージが“COMMAND.COM”によって管理される外部コマンドやバッチ・コマンドにおいて表示される可能性があることを示します。

A. 1 入力待ちメッセージ

Abort, Retry, Ignore? ■

※COMMAND

意味： 中止<A>、再試行<R>、無視<I> ?

内容： ディスク入出力中にエラーが発生した場合に他のエラー・メッセージとともに表示される、そのエラーに対する処置を求めるメッセージです。

対処： “A”、“R”または“I”を入力します。それぞれの意味は次の通りです。

Abort : その処理を中止し、コマンド入力待ちに戻ります。

Retry : その処理をもう一度試みます。

Ignore : そのエラーを無視し、次の処理に移ります。

Are you sure (Y/N)? ■

※DEL

意味： よろしいですか <Y/N> ?

内容： “del” コマンドでファイル名の指定に “*. *” を用いると（全部のファイルを削除という意味になる）表示されます。

対処： 全ファイルを削除してもよいのなら “Y” を押します。 “N” を押すと、削除せずにコマンド入力待ちに戻ります。

Boot error
Press any key for retry

※MSX-DOS起動時

意味： MSX-DOSを起動できません。

任意のキーを押し、もう一度起動してください。

内容： ドライブAに挿入されているディスクに “MSXDOS.SYS” が入っていないためMSX-DOSを起動することができません。

対処： “MSXDOS.SYS” が記録されているディスクをドライブAに挿入したのち、任意のキーを押してもう一度起動してください。

Disk error reading drive <ドライブ番号>
Abort, Retry, Ignore? ■

Disk error writing drive <ドライブ番号>
Abort, Retry, Ignore? ■

意味： ディスクにトラブルが発生しました。

内容： ディスクにキズがあるなどの物理的な事故があった場合、あるいはドライブに故障がある場合にこのエラーが発生します。

対処： 何度か “R” を入力しそれでもエラーが繰り返されるようであれば、そのディスクは使用不能です。新しいディスクでその処理を行ってください。それでもエラーが発生したらドライブの故障と考えられます。

Insert disk with batch file
and strike any key when ready ■

※COMMAND（バッチ・コマンド）

- 意味： バッチ・ファイルのあるディスクをドライブに入れてください。
準備ができたら任意のキーを押してください。
- 内容： バッチ・コマンド実行中に、そのバッチ・ファイルがあるディスクを引き抜いてしまった場合に表示されます。
- 対処： 実行中のバッチ・ファイルがあるディスクをもとのドライブに挿入してから任意のキーを押します。

Insert diskette for drive <ドライブ番号>
and strike a key when ready ■

- ※COMMAND, COPY, DEL, DIR, FORMAT, REN, TYPE
- 意味： ディスクをドライブ<ドライブ番号>に入れてください。
準備ができたら任意のキーを押してください。
- 内容： 仮想ドライブ機能が動作しているとき、複数のドライブに対して処理を行うコマンドを実行すると表示されます。
- 対処： 第2章第6節第4項「仮想ドライブ機能」を参照してください。

Insert DOS disk in default drive
and strike any key when ready ■

- ※COMMAND
- 意味： COMMAND.COMのあるディスクをデフォルト・ドライブに入れてください。
準備ができたら任意のキーを押してください。
- 内容： 外部コマンド終了時に、デフォルト・ドライブに挿入されているディスクに“COMMAND.COM”が存在しないと、これが表示されることがあります。またMSX-DOS起動時に、ドライブAに挿入されているディスクに“MSXDOS.SYS”だけしか記録されておらず、“COMMAND.COM”がない場合もこのメッセージが表示されます。
- 対処： “COMMAND.COM”が入ったディスクをデフォルト・ドライブに挿入してから任意のキーを押します。

Invalid date
Enter new date ■

- ※DATE
- 意味： 日付の指定が違います。

新しい日付を入力してください。

内 容： 日付の指定に許されていない値を用いると表示されます。

対 処： 正しい値をもう一度入力します。

Invalid time

Enter new time ■

※TIME

意 味： 時刻の指定が違います。

新しい時刻を入力してください。

内 容： 時刻の指定に許されていない値や区切り記号を用いると表示されます。

対 処： 正しい値をもう一度入力します。

Not ready error reading drive <ドライブ番号>

Abort, Retry, Ignore? ■

意 味： ドライブ<ドライブ番号>は読み込む準備ができていません。

内 容： ディスクが正しくドライブに挿入されていない場合に表示されます。

対 処： ディスクが正しくドライブに挿入されているかどうかを確認し、“R”を押します。

Not ready error writing drive <ドライブ番号>

Abort, Retry, Ignore? ■

意 味： ドライブ<ドライブ番号>に書き込む準備ができていません。

内 容： ディスクに書き込み中に誤ってそのディスクを引き抜いてしまった場合や、ドライブに故障があった場合に表示されます。

対 処： i. ディスクを引き抜いてしまった場合はそのディスクを挿入し“R”を入力します。

ii. ドライブの故障の場合はそれを回復してからもう一度その処理を試みます。

Strike a key when ready ■

※DIR/P, FORMAT, PAUSE

意 味： 準備ができたなら任意のキーを押してください。

内 容： 一時停止した画面の表示またはバッチ・コマンドの実行を再開してもよいかどうかをMSX-DOSが知りたいときに表示されます。また、“format” コマンドでは、ディスクの準備ができたかどうかをMSX-DOSが知りたいときに表示されます。

対 処： 必要な処置をとってから任意のキーを押します。

Terminate batch file (Y/N)? ■

※COMMAND (バッチ・コマンド)

意 味： バッチ処理を中止しますか (Y/N) ?

内 容： バッチ処理中にCTRL+Cを押すと表示されます。

対 処： バッチ・コマンドを中止する場合は“Y”を、続行する場合は“N”を入力します。“N”を入力押した場合、バッチ・コマンドは次のコマンド行から続行されます。

Write protect error reading drive <ドライブ番号>

Abort, Retry, Ignore? ■

意 味： ドライブ<ドライブ番号>からの読み込み中にライト・プロテクト・エラーが発生しました。

内 容： このエラーはディスク・インターフェイス・カートリッジに異常がある場合にのみ発生します。

対 処： ディスク・インターフェイス・カートリッジあるいはドライブを点検する必要があります。

Write protect error writing drive <ドライブ番号>

Abort, Retry, Ignore? ■

意 味： ドライブ<ドライブ番号>に対して書き込みができません。

内 容： 書き込みの対象となっているディスクが書き込み禁止になっている (ライト・プロテクト・ノッチが書き込み不可側にセットされている) 場合にこのエラーが発生します。

対 処： i. ライト・プロテクト・ノッチを書き込み可にセットしなおしてから“R”を入力します。

ii. “A”を入力して処理を中止し、ライト・プロテクト・ノッチを書き込

み可にセットしてから、もう一度処理を行います。

注 意： “R”を入力した時には、けっして他のディスクを入れないでください。

(注) 以下のメッセージも入力待ちメッセージですが、ここでは解説を省略します。第5章「コマンド一覧」のそれぞれのコマンドの項を参照してください。

```
Current date is www yy-mm-dd
Enter new date: ■
```

※DATE

```
Current time is hh:mm:ss.tt{a|p}
Enter new time: ■
```

※TIME

A. 2 エラー・メッセージ

```
Bad command or file name
```

※COMMAND

意 味： コマンドまたはファイル名が違います。

内 容： 入力した外部コマンド名あるいはバッチ・ファイル名が指定したドライブ上にない場合や、コマンド名、ファイル名の入力を間違えた場合にこのエラーが発生します。

- 対 処：
- i. ドライブ指定で、そのファイルが記録されているディスクが挿入されているドライブを指定します。
 - ii. そのファイルが記録されているディスクをドライブに挿入し、そのドライブ名を指定します。
 - iii. 正しいファイル名を入力します。

```
Bad FAT, drive <ドライブ番号>
```

意 味： FATが破壊されています。

- 内 容： F A T を読み出すことができなかった時に発生します。また未フォーマットのディスクを使用した場合にも、このエラーになる可能性があります。
- 対 処： 致命的なエラーで、そのディスクに記録されているファイルはすべて破壊されていると考えられます。このエラーが出た場合は、読み出せるファイルがあれば（“t y p e” コマンド、“c o p y” コマンドを使って確認する）、別のディスクにコピーして救済してください。エラーが起こったディスクはフォーマットしなおす必要があります。

B a d p a r a m e t e r

※F O R M A T

意 味： パラメータが違います。

内 容： “f o r m a t” コマンドがフォーマットの様式を選択するようになっていた場合に、許されていない値を入力するとこのエラーが発生します。

対 処： “f o r m a t” コマンドをもう一度起動し、正しい値を入力してください。

D i s k e r r o r

※F O R M A T

意 味： ディスクにエラーが発生しました。

内 容： “f o r m a t” コマンド実行中にディスク入出力関係のなんらかのエラーが発生しました。

対 処： 他のコマンドを実行して具体的にどのようなエラーなのかを調べ、それに対処します。

F i l e c a n n o t b e c o p i e d o n t o i t s e l f

※C O P Y

意 味： ファイルをそれ自身にコピーできません。

内 容： ファイルをそれ自身が記録されているディスクにコピーしようとする、このエラーが発生します。たとえば、次のようなコマンドを投入した場合などです。

- i. A>copy file.ext
- ii. A>copy file.ext a:
- iii. A>copy *. * a:

- 対 処： i. 同じディスクにコピーしたい場合は違うファイル名でコピーします。
ii. 違うディスクにコピーします。

File creation error

※COPY

意 味： ファイルを作れません。

内 容：すでに記録可能な総ファイル数まで使用されているディスク上に、新しくファイルを作成しようとするこのエラーが発生します。

対 処： 新しいディスクを使用してください。

File not found

※DIR, TYPE, DEL, COPY

意 味： ファイルが見つかりません。

内 容： <ファイルスペック>で指定したファイルが、指定したドライブに存在しない場合や、ファイル名の入力を間違えた場合にこのエラーが発生します。

- 対 処： i. そのファイルが記録されているドライブを指定します。
ii. そのファイルが記録されているディスクをドライブに入れます。
iii. 正しいファイル名を入力します。

Insufficient disk space

※COMMAND, COPY

意 味： ディスクの容量が足りません。

内 容： ファイルを作るため、あるいはコピーするために十分な容量がディスクに残っていない場合にこのエラーが発生します。

対 処： 新しいディスクあるいは充分に空きのあるディスクを用いてその処理を行います。

Invalid drive specification

※COMMAND, DIR, TYPE, REN, DEL, COPY

意 味： ドライブの指定が違います。

内 容： ドライブ指定で、使用することができないドライブを指定した場合にこのエラーが発生します。たとえば、ドライブBまでしか使用できないのにドライブ

Cを指定した場合などです。

対 処： 正しいドライブ指定を行います。

Invalid parameter

※MODE、VERIFY

意 味： パラメータが違います。

内 容： “mode” コマンドでは許されている値の範囲を越えて指定した場合、“verify” コマンドでは“on”、“off”以外のものを指定した場合にこのエラーが発生します。

対 処： 正しいパラメータを入力します。

No enough memory

※FORMAT、MSX-DOS起動に関するエラー

意 味： メモリが足りません。

内 容： ディスクが動作するために必要なメイン・メモリがコンピュータに内蔵されていない場合や、“format” コマンドを実行するために十分なメモリがない場合にこのエラーが発生します。

対 処： i. 増設RAMカートリッジにより、メモリを増設する必要があります。
ii. “format” コマンドでこのエラーが起こった時は接続ドライブの台数が多すぎることを考えられるので、その場合は“format” コマンドを実行するときだけドライブ数を少なくします。

Program too big to fit in memory

※COMMAND

意 味： プログラムが大き過ぎて、メモリに入りません。

内 容： 使用できるメモリより大きいプログラムを実行しようとした場合にこのエラーが発生します。

対 処： i. 接続しているディスクの数を減して、使用できるメモリを増やします。
ii. 大きさが小さくなるように、プログラムを変更します。

R e n a m e e r r o r

※REN

意 味： このファイル名変更は間違っています。

内 容： ファイル名を変更する際に、新しいファイル名としてすでにそのディスクに存在するファイル名を指定した場合にこのエラーが発生します。

対 処： d i r コマンドでどのようなファイルが記録されているかを確認し、そのディスクにはないファイル名を指定します。

U n s u p p o r t e d m e d i a t y p e e r r o r r e a d i n g d r i v e <ドライブ番号>

意 味： <ドライブ番号>のディスクはこのドライブでは扱えません。

内 容： そのドライブで扱うことができない種類のディスクに対して入出力しようとした場合にこのエラーが発生します。このとき自動的にA b o r t しします。

対 処： そのドライブで扱えるタイプのディスク以外は使用しないでください。ただしそのディスクがフォーマットされていない可能性もあります。

W r i t e e r r o r

※COPY

意 味： 書き込みができません。

内 容： なんらかの理由で書き込み中のファイルをM S X - D O Sが見失った場合にこのエラーが発生します。

対 処： もう一度最初からやり直します。

用語解説

アスキー

ASCII

アメリカ規格協会 (ANSI) が制定した情報交換用コード (American Standard Code for Information Interchange)、またはこのコード表に記載されている文字をいいます。

アスキー・ファイル

アスキーで書かれたファイルをいいます。

エスケープ・シーケンス

周辺機器などハードウェアの制御に用いられる信号で、エスケープ・コード (ASCII の 1BH) とそれに続く文字で構成されるのでこの名があります。

カーソル

ディスプレイで、キーボードから入力された文字や数字が表示される位置を示す印をいいます。

コントロール・シーケンス

ハードウェアの制御に用いられる信号の総称で、エスケープ・シーケンス、コントロール・コードなどがこれに含まれます。

バイナリ

2進数をいいます。

バイナリ・ファイル

2進数で書かれたファイルをいい、通常 “.COM” という拡張子を持っています。

パラメータ

ファイル名やスイッチなど、ある処理を行うための情報の総称です。引数や変数をいうこともあります。

ファンクション・キー

ある特定の機能を割り当てられているキーをいいます。パーソナル・コンピュータでは、プログラマブル・ファンクション・キーを、単にファンクション・キーと呼んでいます。

フォーム・フィード・コード

コントロール・コードの一つで、画面消去や改ページを行うコードです。

ベル・コード

コントロール・コードの一つで、コンピュータ内蔵のベルを鳴らすコードです。

マスター・ディスク

普通、購入したソフトウェアが記録されているディスクを指します。

EOF

エンド・オブ・ファイル・マークをいい、アスキーの1AHです。キーボードからはCTRL+Zで入力することができます。

索引

ア行	
アスキー	30
アスキー・ファイル	30 31 47
アスタリスク	18 19
アセンブラ	18 56
アプリケーション・プログラム	17
エスケープ・シーケンス	47
エラー・メッセージ	9 64
カ行	
仮想ドライブ機能	10 11 61
カーソル	2 5 7
外部記憶装置	16
外部コマンド	22 23 24 59 61 64
拡張子	17 18 19 20 21 23 24 31 33 36 37 44
キーボード	2 21 50 55
クエスチョン・マーク	19
コマンド	1 2 6 7 10 16 20 21 23 24 26 29 36 38 42 44 49 52 54 55 59
コマンド行	1 6 22 23 24 49 50 51 52 53 57 63
コマンド・ファイル	17 22 23 56
コロソ	20 46
コントロール・シーケンス	47
コンバイラ	23
サ行	
システム・ディスク	4 7 8 23
スラッシュ	35
セクタ	40
タ行	
データ・レコーダ	16
ディスク・インターフェイス	
・カートリッジ	10 11 13 14 63
ディスク・ファイル	16 20 21
ディスプレイ	2 5 11 21 37 39 41 42 47 59
ディレクトリ	16 26 47
デバイス・ファイル	20 21

デフォルト・ドライブ	6	20	23	24	29	31	36	37	44	61
テンプレート	49	50	51	52						
特殊キー	1	49	51	54						
トラック	3	40								
ドライブ指定	20	21	22	23	24	29	37	44	64	66
ドライブ番号	11	60	61	62	63	64	66			
ナ行										
内部コマンド	22	27	28	34	36	37	44	64	66	
入力待ちメッセージ	59	64								
ハ行										
ハードウェア	10	12								
ハイフン	35									
バイト	3									
バイナリ	30									
バイナリ・ファイル	30	47								
バッチ・コマンド	22	24	26	42	59	61	63			
バッチ処理	1	22	63							
バッチ・ファイル	17	24	26	55	56	57	61	64		
パラメータ	2	22	31	34	35	37	38	42	44	45
	46	52	55	56						
ファイルスベック	1	20	27	28	29	30	36	37	38	47
	48									
ファイル名	1	16	17	18	19	20	21	23	26	29
	31	37	38	44	55	57	60	64	66	
ファンクション・キー	27									
フォーム・フィールド・コード	47									
複写	28	30								
プリンタ	4	16	21	54						
フロッピー・ディスク・ドライブ	16									
プログラミング言語	22									
プロンプト	6	7	22							
ベル・コード	47									
マ行										
マイクロ・フロッピー・ディスク	3									
マスター・ディスク	7	9								
マニュアル	7	11								
メッセージ	1	7	9	11	33	34	35	36	39	42
	45	46	59	61	64					
メモリ	5	22	27	67						
ラ行										

[illegible]

mode	23	26	37	38	41	67				
MSX	4	5	10	12	13	14	15	35	41	57
MSX2	10	41								
MSXDOS.SYS	5	17	56	57	60	61				
P										
pause	23	24	26	42	62					
R										
ren	23	26	27	44	61	66	68			
rename	23	26	44	68						
rem	1	23	24	26	42	43				
T										
time	10	23	26	27	45	46	62	64		
type	23	24	26	27	47	52	53	55	61	65
	66	68								
V										
verify	23	26	27	48	67					

第3部
プログラマーズ・ガイド

第3部 プログラマーズ・ガイド 目次

第1章 プログラムの実行環境

1.1 プログラムの起動から終了まで.....	1
1.1.1 プログラムの起動.....	1
1.1.2 プログラムの実行.....	3
1.1.3 プログラムの終了.....	4
1.2 システム・コール.....	4
1.2.1 周辺装置との入出力.....	4
1.2.2 ファイル・アクセス.....	5
1.2.3 環境設定.....	9

第2章 システム・コールの使用法

システム・コール解説.....	10
00 システム・リセット.....	13
0C バージョン番号の取り出し.....	13
01 コンソール1文字入力.....	13
02 コンソール1文字出力.....	13
03 補助入力装置1文字入力.....	14
04 補助出力装置1文字出力.....	14
05 プリンタ1文字出力.....	14
06 コンソール直接入出力.....	14
07 コンソール直接入力.....	15
08 コンソール1文字入力(エコー無)	15
09 コンソール文字列出力.....	15
0A コンソール1行入力.....	16
0B コンソール入力チェック.....	16
0D ディスク・リセット.....	17
0E デフォルト・ドライブの設定.....	17
0F ファイルのオープン.....	17
10 ファイルのクローズ.....	17
11 ファイルの検索(最初の一致) ...	18
12 ファイルの検索(後続の一致) ...	18
13 ファイルの削除.....	18
14 シーケンシャル読み出し.....	19
15 シーケンシャル書き込み.....	19
16 ファイルの作成.....	19
17 ファイル名の変更.....	20
18 ログイン・ベクトルの獲得.....	20
19 デフォルト・ドライブの獲得.....	20
1A 転送アドレスの設定.....	21
21 ランダム読み出し.....	21
22 ランダム書き込み.....	21
23 ファイル・サイズの獲得.....	21
24 ランダム・レコードの設定.....	22
28 ゼロ書き込みを伴うランダム書き込み	22
26 ランダム・ブロック書き込み.....	22
27 ランダム・ブロック読み出し.....	23
2A 日付の獲得.....	24
2B 日付の設定.....	24
2C 時刻の獲得.....	24
2D 時刻の設定.....	24
2E ベリファイ・フラグの設定.....	25

第3章 ディスクファイルの構造

3.1 ディスクファイルの構造.....	26
3.2 ディスクアクセスのためのシステム・コール.....	34

※索引.....	36
----------	----

第1章 プログラムの実行環境

1. 1 プログラムの起動から終了まで

MSX-DOSは、外部コマンドの形で、コマンドを自由に追加／変更していける、拡張性に富んだオペレーティング・システムで、アプリケーション・プログラムやユーザーの作ったプログラムも、簡単にコマンドとして実行できるようになっています。

1.1.1 プログラムの起動

MSX-DOSの基本的な動作は、コマンド行を入力し実行させる、という処理の繰り返しです。この時、コマンド行の入力からコマンドの解釈実行までの部分を受け持つのがコマンド・プロセッサです。外部コマンド（アプリケーション・プログラム）の場合、コマンド・プロセッサは、それを以下のようにして実行に移します。

- i. コマンド行のパラメータ部分について、その長さを80H番地に、実際の文字列を81H番地以降に格納する。さらに最初の2つのパラメータをファイル名とみなし、それをFCBの形式でそれぞれ5CH番地および6CH番地以降に格納する。（FCBについては、“ファイル・アクセス”の項で詳しく説明します）
- ii. プログラムを100H番地以降に読み込み、100H番地にジャンプする。

外部コマンドの実体は、アセンブラやコンパイラなどを利用して作られたコマンド名＋拡張子“COM”を持つディスク上のファイルです。コマンド・プロセッサは外部コマンドをディスク上に見つけると、それをメモリ上にロードして実行します。このファイルの内容はメモリの100H番地からロードして、そのまま実行できるような形式になった機械語のプログラムです。

MSX-DOSでは、システムとプログラムとの間でデータ等を受け渡しするために、メモリの0～FFH番地を固定の目的に使っています。これを、システム・スクラッチエリアと呼んでいます。コマンド・プロセッサはプログラムにコマンド行のパラメータを渡すために、システム・スクラッチエリアの設定を行います。このために2つの方法が使われます。5CH番地または6CH番地はそのままファイルのアクセスに使えるような形式になっていて(*1)、ファイル名をパラメータとするようなプログラムではたいていこちらの方法でパラメータを受けとります。80H番地の方には全パラメータがそのまま入っていますからファイル名以外をパラメータとしたり、あるいは3つ以上のファイル名を扱うようなプログラムではこの方法を使用します。

(*1) 両FCBの先頭アドレスは16バイトしか離れていないので、完全なFCBとして使用できるのはどちらか片方だけです。

0000H	JP NBOOT		JP BDOS		JP RDSLT	
0010H		JP WRSLT		JP CALSLT		
0020H		JP ENASLT				
0030H	JP CALLF		JP INTRPT			
0040H						
0050H						
0060H					00 Y Y Y	
0070H			D A T	00 00 00 00	01 Z Z Z	
0080H	Z		? ? ?	00 00 00 00	00 00 00 00	
0090H	19	y y y	d a t	b ; z z z z		
	.	* c ; * . w w w				
00F0H						

図 1.1 システム・スクラッチ・エリア

図 1.1 はシステム・スクラッチ・エリアの内容です。これは、

A>test yyy. dat a:zzzz.* b:*. www

というコマンドを入力して、外部コマンド **test** を実行したときの例です。80H 番地以降にはこのコマンドに渡すパラメータとその長さが入っているのがわかります。また、さきほど簡単に説明したように、5CH 番地と 6CH 番地には最初の 2 つのパラメータが形式を整えられて入っているのがわかります。先頭 1 バイトには、ドライブが無指定であった場合は 0、“A:” であれば 1、“B:” なら 2・・・という値が入ります。小文字が大文字に変換されていたり、“*” が複数の“?” に展開されていることにも注目してください。ファイル・アクセスには、このような形式になったファイル名を使います。

システム・スクラッチエリアの 5CH~FFH 番地は、このようにしてプログラムにパラメータを渡すために使われますが、プログラムが実行を開始した後は、このエリアをワークとして使うことができます。さきほど述べたように、5CH 番地あるいは 6CH 番地をそのまま FCB として使うことなどの例があります。

1.1.2 プログラムの実行

図1.2は、プログラムが実行を開始したときのメモリ・マップです。0～FFFH番地はシステム・スクラッチエリアで、プログラムその直後の100H番地からロードされています。100H番地から始まり、6～7番地の内容で指し示されているアドレスの1バイト手前までの領域はTPA（Transient Program Area＝一時プログラム領域）と呼ばれ、プログラムで自由に利用してよい領域です。

システム・スクラッチエリアの先頭の方には、いくつかのジャンプ命令が埋め込まれています（図1.1）。プログラムはこれらの決められたアドレスをコールすることで、MSXシステムの持つ各種の機能が実行できます。0番地にあるのがウォーム・スタートのためのエントリ、5番地にあるのが2章で解説するシステム・コールのためのエントリで、この2つはMSX-DOSが持っている機能をプログラムから利用するために用意されています。また、RDSLT、WRSLT、CALSLT、ENASLT、CALLF及びINTRPTは、MSX-BIOSにある同名のファンクションと同じ機能を持っています。INTRPTは割り込み時に使われますが、その他のエントリは、MSX-BIOSと同様に利用できます。MSX-DOS自身もこれらのエントリを使用していますので、0番地～5BH番地（斜線の部分）を使用するとシステム・ダウンにつながります。

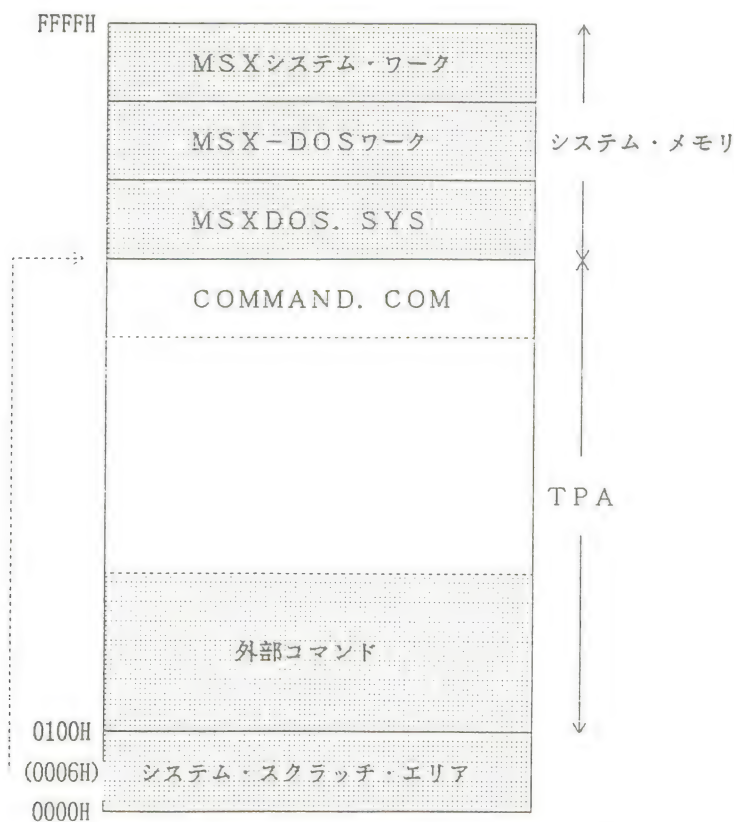


図1.2 外部コマンド実行時のメモリ・マップ

(注) 5 番地はシステム・コールのためのジャンプ命令です。6～7 番地はジャンプ先のアドレスで、これがTPAの上限を示しているということは、ジャンプ先がMSXDOS・SYSの先頭になっているということです。5～7 番地はジャンプ命令であると同時にTPAの上限を示すという2重の働きを持っています。

1.1.3 プログラムの終了

プログラムは実行が終わると、次の3つの方法のいずれかでMSX-DOSをウォーム・スタートしコマンド・レベルに戻ります。

- i. スタック・ポインタを変更していない場合、“RET”を実行する。
- ii. 後述のシステム・コールを使って、“システム・リセット”を行う。
- iii. 0 番地（ウォーム・スタートのためのエントリ）にジャンプする。

コマンド・プロセッサの実体はCOMMAND.COMというプログラムで、TPAの内部に位置しています。コマンド・プロセッサが必要なのは、プログラムの実行が終わって、次のコマンドを入力するときで、プログラムの実行中は、コマンド・プロセッサの機能は必要ありませんから、プログラムでコマンド・プロセッサがロードされている領域を使用（破壊）してもかまわないわけです。このため、ウォーム・スタートの際には、（コマンド・プロセッサがプログラムによって破壊されている可能性があるため）いったんMSXDOS・SYSが制御を受け取ります。MSXDOS・SYSは、チェックサムを用いてCOMMAND.COMが破壊されているかどうかを調べてから、コマンド・プロセッサに制御を渡します。COMMAND.COMが破壊されていた場合は、再びディスクからロードしますが、破壊されていない場合には、ディスクからのロードは行われませんので、コマンド・レベルに戻るのが早くなります。

こうして、再びコマンド・プロセッサが次のコマンド行の入力待ちに入ります。

1. 2 システム・コール

1.2.1 周辺装置との入出力

MSX-DOSの持つ機能は、ディスク上のファイルを扱うものだけではありません。それだけでは、オペレーティング・システムの持つ重要な機能の1つであるユーザー・インターフェースの機能が限られてしまいます。そのため、MSX-DOSでは、コンソール（スクリーン、キーボード）やプリンタなどの周辺装置の制御を行う機能を豊富に用意しています。これらの装置は、基本的には1文字単位で入出力を行うものですが、コンソールに関しては、各種のオプションを持った複数の入出力機能に加えて、行入力や行出力の機能も用意しています。

1.2.2 ファイル・アクセス

ファイル・アクセスの特徴は、データの位置をディスク上のアドレスのような具体的数値で表現するのではなく、“名前”を用いて指定できるという点にあります。MSX-DOSはディスク上に書かれたディレクトリ等の情報をもとに、ファイルのデータがどこに存在しているかということを常に把握しています。目的のデータがディスク上のどんなアドレスに存在しているか、ということはすべてMSX-DOSにまかせ、プログラムはただファイル名を指示するだけでそれをアクセスできるのです。

● FCB (ファイル・コントロール・ブロック)

どのファイルにアクセスするか、ということを指示するために、MSX-DOSでは、FCB (ファイル・コントロール・ブロック) を用います。FCBは、ファイルを扱う際必要となる情報を格納しておく領域で、ひとつのファイルを扱うごとにひとつ、図1.3で示すようなメモリ・エリアをプログラムで用意します。FCBはプログラムで許された範囲でメモリ上のどこに置いても構いませんが、MSX-DOSの機能を活かすため、5CH番地がしばしば用いられます。

● ファイルのオープン

FCBを用いてファイルの入出力を行うには、最初にファイルを“オープン”する手続きが必要です。“オープン”とは、ファイル名フィールドだけが設定された不完全なFCBを、ディスク上に記された情報を用いて、完全なFCBに変換することを意味しています。図1.3、図1.4に“オープンされていないFCB”と“オープンされたFCB”の違いを示します。

● ファイルのクローズ

ファイルをオープンして書き込みを行った場合、それに伴って、ファイルサイズを始めとするFCBの各フィールドの内容も変更を受けます。この更新されたFCBの情報をディレクトリ領域に戻しておかないと、次回ファイルをアクセスする際に、ディレクトリの情報と実際のファイルの内容が食い違ってしまいます。更新されたFCBの情報をディレクトリに戻すというこの操作が、ファイルのクローズです。

● DTA (ディスク転送アドレス)

ディスクの入出力では、基本的には文字単位の入出力である周辺装置と違って、一度に複数のバイトを読み書きします。そのため、ディスクの入出力には“バッファ”が必要になります。MSX-DOSでは、このバッファとして使用されるメモリ領域の先頭アドレスをDTA (Disk Transfer Address=ディスク転送アドレス) と呼んでおり、ディスクの入出力では、常にDTAをバッファとして使用します。DTAの値は、0080Hに初期化されていますが、任意のアドレスに設定し直すことが可能です。

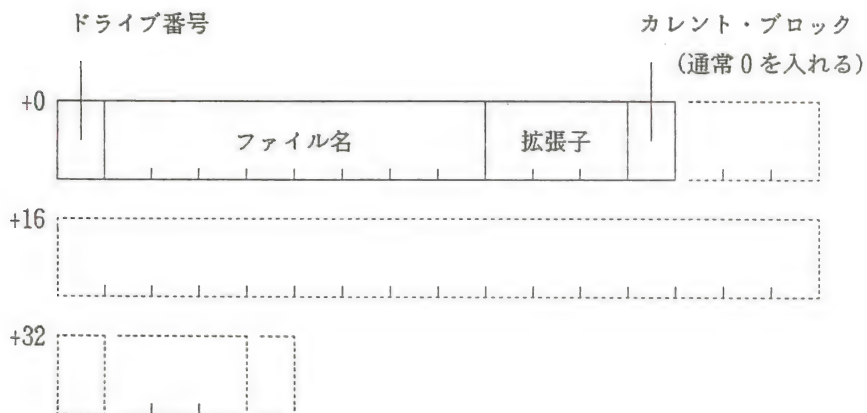


図 1.3 FCB (オープン前)

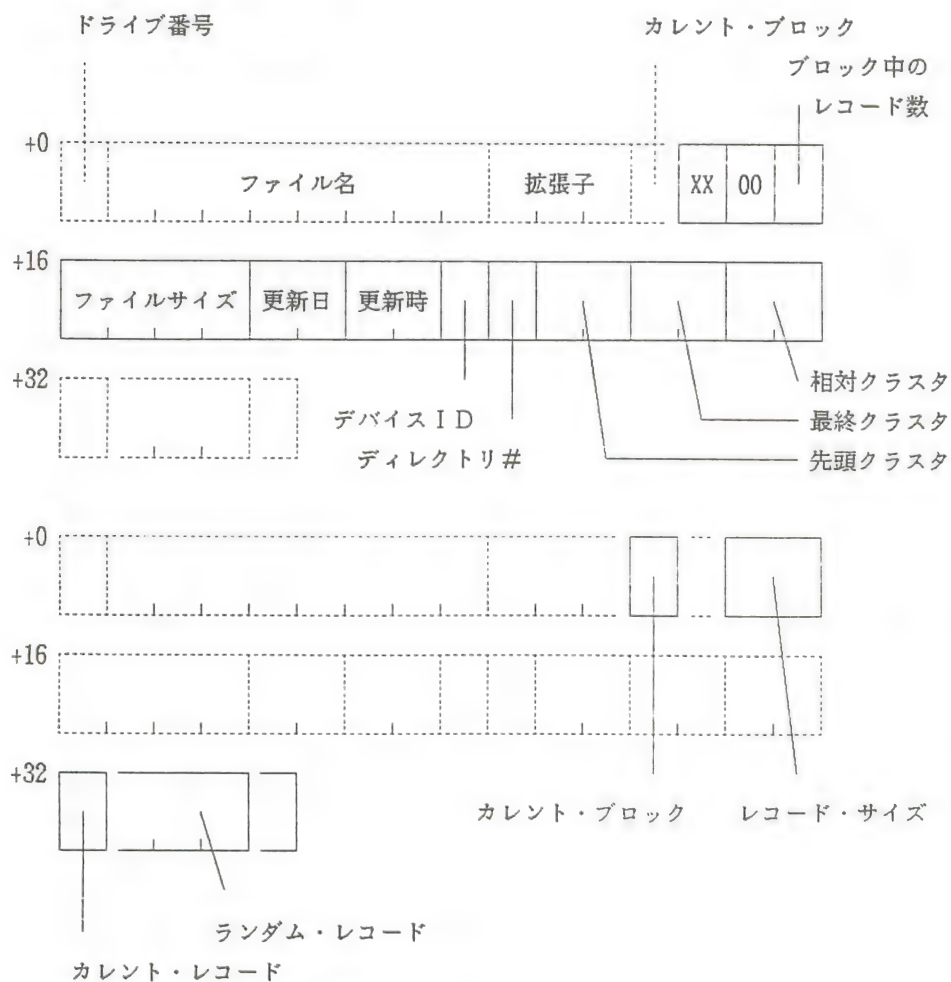


図 1.4 FCB (オープン後)

*ドライブ番号(+0)

ファイルの存在するディスクドライブを示す

(0→デフォルト・ドライブ、1→A、2→B、… 8→H)

*ファイル名本体(+1~8)

最大8文字まで指定可能。8文字に足りない部分はスペース(20H)で埋める。

*拡張子(+9~11)

最大3文字まで指定可能。3文字に足りない部分はスペース(20H)で埋める。

*カレントブロック(+12、+14)

シーケンシャル・アクセスの際、参照中のブロック番号を示す。

(ファンクション 14H, 15H, 24H)

*ブロック中のレコード数(+15)

ブロックに含まれるレコード数が設定される。最終ブロック以外は常に128。

*レコードサイズ(+14~15)

レコードサイズをバイト数で指定する。(ファンクション 26H, 27H)

*ファイルサイズ(+16~19)

ファイルの大きさがバイト単位で設定される。

*日付(+20~21)

最後にファイルに書き込みを行った日付が設定される。

*時刻(+22~23)

最後にファイルに書き込みを行った時刻が設定される。

*デバイスID(+24)

周辺装置/ディスクファイルの区別が設定される。

*ディレクトリ・ロケーション(+25)

何番めのディレクトリ・エントリに該当するファイルであるかが設定される。

*先頭クラスタ(+26~27)

ファイルの先頭のクラスタ番号が設定される。

*最終アクセスクラスタ(+28~29)

最後にアクセスされたクラスタ番号が設定される。

*最終アクセスクラスタの先頭クラスタからの相対位置(+30~31)

最後にアクセスされたクラスタのファイル内での相対番号が設定される。

*カレントレコード(+32)

シーケンシャルアクセスの際、参照中のレコード番号を示す。(14H, 15H, 24H)

*ランダムレコード(+33~36)

ランダムアクセスおよびランダムブロックアクセスの際、アクセスしたいレコードを指定する。レコードサイズが1~63であると+33~36の4バイトを使用する。

レコードサイズが64以上の場合には+33~35の3バイトを使用する。

(ファンクション 21H, 22H, 23H, 24H, 26H, 27H, 28H)

● レコード

ファイルの大きさは、ディスク容量の許す限りいくらでも（最大1 Gバイト）大きくなれますから、メモリに全体が読み込めるといったことは期待できません。そのためファイルの入出力では、ファイルをレコードと呼ばれる適当な大きさのデータ単位に分割し、レコード単位で使って読み書きする方法がとられています。

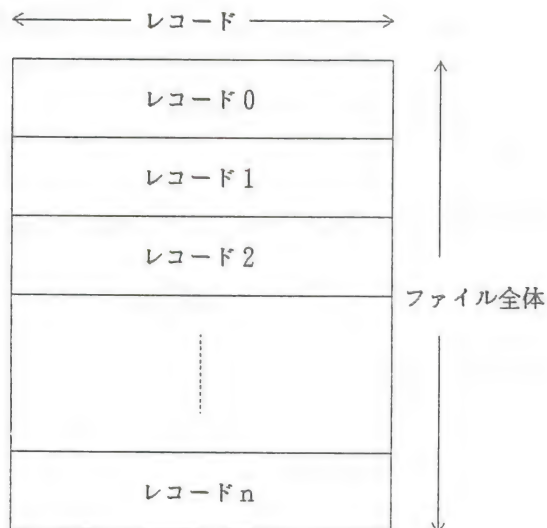


図1.5 ファイルとレコード

従って、ファイルのアクセスを行うためには、さらに、どのレコードをアクセスするかという指示が必要になってきます。このレコードの指示方法はシステム・コールの種類により、大きく2つのグループに分けられます。ひとつのグループはCP/Mとの互換性を保つために用意された入出力で、もう1つのグループは、MSX-DOS独自のランダム・ブロック入出力です。この2つは、FCBの使用方法が異なりますので、同じFCBに対して同時に使用することはできません。

● CP/M方式のレコード・アクセス

CP/Mとの互換性を保つため、MSX-DOSではCP/Mと同様のファイルアクセス方法をサポートしています。CP/Mでは、2つの全く異なるアクセス方式があり、またレコードサイズは128に固定されています。

・シーケンシャルアクセス

(カレントレコード+カレントブロックによるレコード管理)

1つめの方法が、“カレントレコード”と“カレントブロック”で管理されるシーケンシャルアクセスです。ファイルのアクセスは常に先頭から順（シーケンシャル）に行い、アクセスしたレコード数はFCBのカレントレコード・フィールドでカウントされます。カレントレコード・フィールドの値は128に達すると0にリセットされ、その桁上がりがカレントブロック・フィールドにカウントされます。このアクセス方式では

“ランダムレコード”は使いません。

- ・ランダムアクセス

(ランダムレコードによるレコード管理)

2つめの方法が、“ランダムレコード(3バイト)”で管理されるランダムアクセスです。“ランダムレコード”を変更することで任意の位置のレコードをアクセスすることが可能ですが、別のレコードをアクセスするためには毎回この操作が必要です。

- ランダム・ブロック・アクセス

MSX-DOSには、ランダム・ブロック・アクセスという、大変に有能な入出力方式が存在します。これはMSX-DOS独自のシステム・コールで、次のような特徴を持っています。

- ・レコードサイズを任意に設定できる
- ・複数のレコードを一度にアクセスできる
- ・シーケンシャル／ランダムを任意に切り換えられる

このファイル・アクセスでは、FCBの“レコードサイズ”を用いて、レコードの大きさを指示し、レコード位置を“ランダムレコード”で指示します。レコードサイズの2バイトには任意の値を入れることができます(1～65535)。ファイル全体を1レコードとして扱うことも、1バイトを1レコードとして扱うことも、128バイトを1レコードとして扱うこと(CP/M方式)も可能です。

このアクセス方式では、ランダムレコードのフィールドがアクセスの終わったレコードの次のレコードを指すように常に更新されます。従って、ランダムレコードを一度セットするだけで、任意のレコードからレコードをシーケンシャルに読み書きすることが簡単に行えます。

1.2.3 環境設定

MSX-DOSは、ファイルの内容に変更が加えられた時には、自動的にファイルの最終更新日付／時刻を現在の日付／時刻に書き替えます。これは、MSX-DOS自身には何の意味もない機能ですが、ユーザーがファイルを管理する目的には、非常に有用なものです。このため、MSX-DOSでは、日付／時刻の管理を行っており、これを読み出したり、変更したりするシステム・コールが存在します。

2 章 システムコールの使用法

MSXディスクシステムの持つ入出力機能はBDOSというかたちで、ディスク・カートリッジのROMの中に収められた汎用のサブルーチンにまとめられています。システムコールとは、BDOSを呼び出すためのあらかじめ決められた手順のことで、ユーザーのプログラムはシステムコールを実行することで入出力操作が簡単に実行できるようになります。

システムコールの役割には、大きく分けると次の2つがあります。1つは、基本的な機能をあらかじめ用意することによってプログラムの負担を少なくすること、もう1つは、すべてのプログラムが共通の手順を使用することによって、移植性や汎用性を高めることです。このシステムコールを自由自在に活用することによって、プログラム開発の期間は短縮され、出来上がったプログラムは移植性の高いものとなります。これは、OSを利用した時の最大のメリットのひとつです。

システムコールを実行するには、Z80CPUのCレジスタに決められたファンクション番号を入れ、次のアドレスをコールします。

0005H.....MSX-DOS上の外部コマンド

F37DH.....DISK-BASIC上のマシン語プログラム

たとえば、ファンクション番号が1FHであり、前準備としてAレジスタに00Hをセットすることになっているシステムコールがあるとする、このシステムコールは次のようにして実行します。

LD A, 00H

LD C, 1FH

CALL 0005H (DISK-BASICの場合 0F37DH)

CALL文の後には、戻り値（処理の結果）を受け取ったり、退避していたレジスタを復帰させたりする事後処理が続きます。

DISK-BASICのプログラムからは、CLEAR文で確保した領域にマシン語プログラムを格納し、その開始アドレスをUSR関数で呼び出してやることでシステムコールを利用することができます。

●システムコールの形式

ここでは、以下の書式によってシステムコールの使用法を紹介します。

ファンクション：ファンクション番号

設定：システムコールを行う前にレジスタやメモリ上にセットすべき値

戻り値：システムコールから戻った時にセットされている値

ファンクション：

ファンクション番号とは、システムコールの種類を判別するためのもので、それぞれのシステムコールに対応する番号が決められています。システムコールを実行する時には、このファンクション番号をCレジスタにセットします。

設定：

また、システムコールの前準備として、レジスタやメモリに必要な値をセットしなければならないことがあります。ここではこれを“設定”の部分に示します。

戻り値：

システムコールの結果得られた値は、多くの場合レジスタやメモリの内容にセットされることになります。これがどこにどのようにセットされるかを、ここでは“戻り値”の部分に示します。

ただし、システムコールでは、戻り値がセットされるレジスタ以外にも、ファンクション内での作業に使用され、その内容が破壊されるレジスタがあります（システムコールでは原則としてレジスタのもとの内容は保証していません）。そこで、システムコールを使用する場合には、その前に、破壊されては困るレジスタの内容を、適当な場所（スタックなど）に退避させておくようにしてください。

MSX-DOSのシステムコールは表2.1に示す42個ありますが、これらを次の4つに分類して説明します。

- ・システム・コントロール
- ・周辺装置とのI/O
- ・ファイル・アクセス
- ・環境設定

No	機能	No	機能
00	システム・リセット	13	ファイルの削除
01	コンソール1文字入力	14	シーケンシャル読み出し
02	コンソール1文字出力	15	シーケンシャル書き込み
03	補助入力装置1文字入力	16	ファイルの作成
04	補助出力装置1文字出力	17	ファイル名の変更
05	プリンタ1文字出力	18	ログイン・ベクトルの獲得
06	コンソール直接入出力 (入力待無)	19	デフォルト・ドライブの獲得
07	コンソール直接入力	1A	転送アドレスの設定
08	コンソール1文字入力 (エコー無)	1B	ディスク情報の獲得(*)
09	コンソール文字列出力	21	ランダム読み出し
0A	コンソール1行入力	22	ランダム書き込み
0B	コンソール入力チェック	23	ファイル・サイズの獲得
0C	バージョン番号の取り出し	24	ランダム・レコードの設定
0D	ディスク・リセット	26	ランダム・ブロック書き込み
0E	デフォルト・ドライブの設定	27	ランダム・ブロック読み出し
0F	ファイルのオープン	28	ゼロ書き込みを伴うランダム書き込み
10	ファイルのクローズ	2A	日付の獲得
11	ファイルの検索 (ワイルド・カードに一致する最初 のファイル)	2B	日付の設定
12	ファイルの検索 (ワイルド・カードに一致する後続 のファイル)	2C	時刻の獲得
		2D	時刻の設定
		2E	ベリファイ・フラグの設定
		2F	論理セクタの読み出し(*)
		30	論理セクタの書き込み(*)

(*) 3章で解説。

表 2.1 システム・コール一覧

システムコールのファンクション番号は00Hから30Hまでですが、このなかには以下に示す無効なファンクション番号があります。

1CH～20H、25H、29H

これらの無効なシステムコールを実行した場合には、Aレジスタに00Hがセットされる以外は何も行いません。また、31H以降のファンクション番号でも結果は同じです。

1. システム・コントロール

●システム・リセット

ファンクション: 00H

設定: なし

戻り値: なし

MSX-DOS上のプログラムからコールした場合には、MSX-DOSがウォームスタートし、DOSのコマンドレベルに戻る。MSX DISK-BASIC上のプログラムからコールした場合には、DISK-BASICがウォームスタートし、BASICのコマンドレベルに戻る。この場合、ロードされているプログラムは破壊しない。

●バージョン番号の獲得

ファンクション: 0CH

設定: なし

戻り値: HLレジスタ←0022H

このシステムコールは、CP/Mにおける、さまざまなCP/Mのバージョン番号を獲得するためのものであるが、MSX-DOSの場合には、一律に0022Hが戻される。

2. 周辺装置との入出力

周辺装置の入出力に利用するシステムコールであり、コンソール（画面／キーボード）や外部入出力装置、プリンタなどに関する制御を行います。文字列入力やプリンタ出力などのルーチンを作成するのは、多くのプログラムに必要な定例作業ですが、本節のシステムコールを利用することにより、簡単に汎用的なプログラムを組むことができます。

●コンソール入力

ファンクション: 01H

設定: なし

戻り値: Aレジスタ←コンソールから入力した1文字

入力がない場合（キーボードバッファが空の場合）には、入力待ちを行う。入力された文字はコンソールにエコーバックされる。以下に示すコントロールキャラクタはファンクション内部で処理され入力として扱われない。

Ctrl-C システム・リセット

Ctrl-P プリンタへのエコー開始

Ctrl-N プリンタへのエコー停止

Ctrl-Cを受け付けるとプログラムの実行を中断してMSX-DOSのコマンドレベルに戻る。Ctrl-Pを受け付けると、以降のコンソール出力はすべてプリンターにもエコーされるようになる。この状態は、Ctrl-Nを受け付けることで解除される。

●コンソール出力

ファンクション： 02H

設定： Eレジスタ←出力する文字コード

戻り値： なし

Eレジスタで指定した文字を画面に表示する。この際コンソール入力をチェックし上記3種類のコントロールキャラクタ及びCtrl-Sの処理をおこなう。Ctrl-Sを受け付けると、次に任意のキーを押すまでファンクション内部で入力待ちをおこなうのでプログラムの一時停止ができる。

●補助入力

ファンクション： 03H

設定： なし

戻り値： Aレジスタ←補助入力装置から読み込んだ1文字

補助入力装置がセットされていない場合、常にEOF文字(1AH)を返す。
コンソール入力のチェックを、ファンクション02Hと同様におこなう。

●補助出力

ファンクション： 04H

設定： Eレジスタ←補助出力装置に出力する文字コード

戻り値： なし

補助出力装置がセットされていない場合、何もしない(出力は捨てられる)。
コンソール入力のチェックを、ファンクション02Hと同様におこなう。

●プリンタ出力

ファンクション： 05H

設定： Eレジスタ←プリンタに出力する文字コード

戻り値： なし

コンソール入力のチェックを、ファンクション02Hと同様におこなう。

●直接コンソール入出力

ファンクション: 06H

設定: EレジスタにFFHをセットすれば入力、FFH以外をセットすれば出力となる。EレジスタにFFH以外の値がセットされている場合は、セットされた値を文字コードとみなしてコンソールに出力をする。

戻り値: EレジスタがFFHにセットされていた場合には、入力を実行し、Aレジスタに、結果をセットする。コンソール入力がある場合はその文字コード、無い場合は00Hとなる。EレジスタがFFH以外にセットされていた場合には、戻り値はなし。

入力のエコーバック、コントロールキャラクタの処理は行わない（入力文字として扱われる）。出力のプリンターへのエコーは行わない。

●直接コンソール入力

ファンクション: 07H

設定: なし

戻り値: Aレジスタ←コンソールから入力した1文字

入力のエコーバック、コントロールキャラクタの処理は行わない。

このシステムコールはMSX-DOS用に変更されたものでCP/Mとの互換性はない。

●エコーなしコンソール入力

ファンクション: 08H

設定: なし

戻り値: Aレジスタ←コンソールから入力した1文字

エコーバックは行わない。コントロールキャラクタはファンクション01Hと同様に処理する。

このシステムコールはMSX-DOS用に変更されたものでCP/Mとの互換性はない。

●文字列出力

ファンクション: 09H

設定: DEレジスタ←メモリ上に用意した、コンソールに出力すべき文字列の先頭アドレス

戻り値: なし

文字列の最後には、終端文字として“\$”（24H）を付加しておく。最初にあらわれる“\$”の直前までの文字列をコンソールに出力するので、“\$”を出力することはできない。

コントロールキャラクタのチェックを、ファンクション02Hと同様に行う。

●文字列入力

ファンクション： 0AH

設定： 行バッファ+0 ←最大入力文字数（1～255）（図2.1）

DEレジスタ←行バッファの先頭アドレス

戻り値： 行バッファ+1 ←実際に入力された文字数

行バッファ+2～

←コンソールから入力された文字列

リターンキーの直前までをコンソールからの入力とみなす。最大入力文字数を越えて入力することはできない。このシステムコールによる文字列入力時には、テンプレートによる編集が可能である。また、この際コントロールキャラクタのチェックを行う。

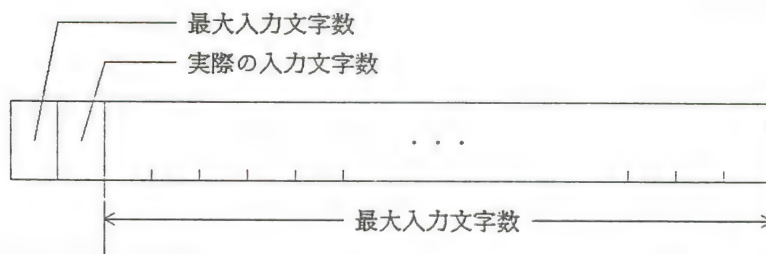


図 2.1 行バッファの構造

●コンソール入力状態のチェック

ファンクション： 0BH

設定： なし

戻り値： コンソール入力がある場合にはFFHを、ない場合には00Hを、Aレジスタにセットする。

入力文字がある場合もその文字は取り出さない、ただしこの際、コントロールキャラクタを、ファンクション02Hと同様にチェックする。入力文字自体は、前述のファンクション01H、またはファンクション08Hで取り出すことができる。

3. ファイルのアクセス

●ディスクリセット

ファンクション: 0DH

設定: なし

戻り値: なし

変更されたのちに、まだディスクに書き込まれていないセクタがあれば、それをディスクに書き込こんだ後、デフォルト・ドライブをAドライブにセットし、ディスク転送アドレスを0080Hにセットする。

●デフォルト・ドライブの設定

ファンクション: 0EH

設定: Eレジスターデフォルト・ドライブ番号

(A=0、B=1、… H=7)

戻り値: なし

FCB等のドライブ番号に0が指定されたときに使用する、デフォルト・ドライブを変更する。接続されていないドライブが指定された場合には、デフォルト・ドライブは変更しない。

●ファイルのオープン

ファンクション: 0FH

設定: DEレジスターオープンされていないFCBの先頭アドレス

戻り値: (オープン成功)

Aレジスター←0

FCB←各フィールドの設定が行われる

(オープン失敗)

Aレジスター←0FFH

FCBがファイル入出力を行うシステムコールで使えるようになる。レコードサイズ、カレント・ブロック、カレント・レコード、ランダム・レコードの各フィールドは使用するファンクションに応じてオープンの後ユーザーが設定する必要がある。

●ファイルのクローズ

ファンクション: 10H

設定: DEレジスターオープンされたFCBの先頭アドレス

戻り値: ファイルのクローズが成功すれば00Hを、失敗すればFFHをAレジスタにセットする。

現在のFCBの内容をディスク上の該当するディレクトリ・エリアに書き込むことによって、ファイルの更新に関する整合性を保つ。ただし、ファイルに対して変更を行っていない場合には、書き込みは行わない。クローズしたFCBはオープンされていないFCBとして再利用してもよいし、あるいは、オープンされたFCBとしてさらに入出力を続けてもよい。

●ファイルの検索（最初の一致）

ファンクション： 11H

設定： DEレジスタ←オープンされていないFCBの先頭アドレス

戻り値： ファイルが見つかった場合は0を、見つからなかった場合は0FFHをAレジスタにセットする。見つかった場合はさらに、DTAで指示される領域にドライブ番号を、それに続く32バイトにそのファイルのディスク上のディレクトリ・エントリをセットする。

ファイル名にはワイルドカード・キャラクタの“?”を使用することができる。たとえば、ファイル名の名前部分に“????????”、拡張子部分に“MAC”という設定を行うと、“MAC”という拡張子をもつファイルのうち、最初にマッチしたファイルのディレクトリ情報を得ることができる。システムコール・レベルではワイルドカード・キャラクタの“*”はサポートしていないので、適当な数の“?”に置き換えなければならない。マッチするすべてのファイルを検索したい場合や、マッチするファイルが1個だけかどうかを知りたい場合には、次に紹介するファンクション12Hを利用する。

DTA領域に設定されたものはオープンされていないFCBとして使用できる。

●ファイルの検索（後続の一致）

ファンクション： 12H

設定： なし

戻り値： ファイルが見つかった場合は0を、見つからなかった場合は0FFHをAレジスタにセットする。見つかった場合はさらに、DTAで指示される領域にドライブ番号を、それに続く32バイトにそのファイルのディスク上のディレクトリ・エントリをセットする。

このシステムコールは、ファンクション11Hのワイルドカード・キャラクタによるファイル名の指定とマッチした複数のファイルを検索する時に使用するもので、単独で使うことはできない。このシステムコールを繰り返し使用して、ファンクション11Hによってマッチした複数のファイルのディレクトリ情報を、1つずつ順番に得ていくことができる。

DTA領域に設定されたものはオープンされていないFCBとして使用できる。

●ファイルの削除

ファンクション： 13H

設定： DEレジスタ←オープンされていないFCBの先頭アドレス
戻り値： 削除が成功した場合は00Hを、失敗した場合はFFHをAレジスタにセットする。

ファイル名にワイルドカード・キャラクタを指定して、複数のファイルを一度に削除することができる。

●シーケンシャルな読み出し

ファンクション： 14H

設定： DEレジスタ←オープンされたFCBの先頭アドレス
FCBのカレントブロック←読み出すブロック番号
FCBのカレントレコード←読み出すレコード番号
戻り値： 読み出しが成功した場合は00Hを、失敗した場合は01HをAレジスタにセットする。成功した場合には、DTA以降の128バイトに読み込んだレコードをセットする。

FCBのカレントブロックとカレントレコードは、読み出し後に次のレコードを指すように自動的に更新される。つまり、連続して読み出しを行う場合には、カレントブロックとカレントレコードをセットする必要はない。

このファンクションが失敗するのは、ファイルの終りに達した時である。

●シーケンシャルな書き込み

ファンクション： 15H

設定： DEレジスタ←オープンされたFCBの先頭アドレス
FCBのカレントブロック←書き込むブロック番号
FCBのカレントレコード←書き込むレコード番号
DTA以降の128バイト←書き込むデータ
戻り値： 書き込みが成功した場合は00Hを、失敗した場合は01HをAレジスタにセットする。

FCBのカレントブロックとカレントレコードは、書き込み後に次のレコードを指すように自動的に更新される。つまり、連続して書き込みを行う場合には、カレントブロックとカレントレコードをセットする必要はない。

このファンクションが失敗するのは、ディスクの残り容量が無くなった時である。

●ファイルの作成

ファンクション： 16H

設定： DEレジスタ←オープンされていないFCBの先頭アドレス。
戻り値： ファイルの作成が成功すれば00Hを、失敗すれば0FFHをAレジスタにセットする。

F C B中のレコードサイズ、カレントブロック、カレントレコード、ランダムレコードの各フィールドについては、これらを本システムコールの実行後に必要に応じてセットしておく必要がある。

このファンクションが失敗するのは、ファイル名が正しくセットされていない時、あるいは、ディレクトリに空きがない時である。

●ファイル名の変更

ファンクション： 17H

設定： DEレジスタ←オープンされていないF C Bの先頭アドレス。

F C B + 0 ←旧ファイル名

F C B + 16 ←新ファイル名

戻り値： ファイル名の変更が成功すれば0を、失敗すれば0 F F HをAレジスタにセットする。

新旧ファイル名に、ワイルドカード・キャラクタを使用できる。たとえば、旧ファイル名に“???????REL”を、新ファイル名に“???????LIB”を指定すれば、“REL”という拡張子をもつすべてのファイルについて、拡張子を“LIB”に変更することができる。

●ログインベクトルの獲得

ファンクション： 18H

設定： なし

戻り値： HLレジスタ←オンラインドライブ情報

MSXには最大8台までのディスクドライブが接続できるが、オンラインドライブとはそのうち、MSXに実際に接続されているドライブをさす。このシステムコールを実行すると、各ドライブがオンラインであるかどうかにより、結果を図2.2のようにHLレジスタに入れて返す。それぞれのビットが1ならば対応するドライブはオンラインであり、0ならばそうでないことを示す。MSX-DOSの場合、Hレジスタは常に0である。

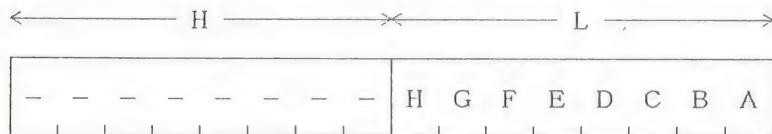


図2.2 ログイン・ベクトル

●デフォルト・ドライブ番号の獲得

ファンクション： 19H

設定： なし

戻り値： Aレジスタ←デフォルト・ドライブ番号
(A=0、B=1、… H=7)

●転送先アドレスの設定

ファンクション： 1AH

設定： DEレジスタ←設定する転送先アドレス (DTAアドレス)

戻り値： なし

DTAアドレスは、システムリセット時に0080Hに初期化されるが、このシステムコールを用いることによって、任意のアドレスに設定し直すことができる。

●ランダムな読み出し

ファンクション： 21H

設定： DEレジスタ←オープンされたFCBの先頭アドレス
FCBのランダムレコード←読み出すレコード番号

戻り値： 読み出しが成功すれば00Hを、失敗すれば01HをAレジスタにセットする。読み出しが成功した場合には、DTA以降の128バイトに読み出したレコードをセットする。

レコードの大きさは128バイト固定長

●ランダムな書き込み

ファンクション： 22H

設定： DEレジスタ←オープンされたFCBの先頭アドレス
FCBのランダムレコード←書き込むレコード番号
DTA以降の128バイト←書き込むデータ

戻り値： 書き込みが成功すれば00Hを、失敗すれば01HをAレジスタにセットする。

レコードの大きさは128バイト固定長

●ファイルサイズの獲得

ファンクション： 23H

設定： DEレジスタ←オープンされたFCBの先頭アドレス

戻り値： ファイルサイズの獲得に成功すれば00Hを、失敗すれば0FFHをAレジスタにセットする。獲得が成功すれば、FCBのランダムレコード・フィールドに、指定されたファイルの128バイト単位のサイズをセットする。

ファイルのサイズは128バイト単位で計算される。つまり、200バイトのファイル

であれば2が、257バイトのファイルであれば3がセットされる。

●ランダムレコード・フィールドの設定

ファンクション: 24H

設定: DEレジスタ←オープンされたFCBの先頭アドレス

FCBのカレントブロック←目的のブロック

FCBのカレントレコード←目的のレコード

戻り値: ランダムレコード・フィールドに、指定されたFCBのカレントブロック・フィールドとカレントレコード・フィールドから計算したカレントレコード・ポジションをセットする。

●ゼロ書き込みを伴う、ランダムな書き込み

ファンクション: 28H

設定: DEレジスタ←オープンされたFCBの先頭アドレス

FCBのランダムレコード←書き込むレコード番号

DTA以降の128バイト←書き込むレコード

戻り値: 書き込みが成功すれば00Hを、失敗すれば01HをAレジスタにセットする。

レコードの大きさは128バイト固定長

本システムコールでは、ファイルサイズより大きなレコード番号が指定された場合、ファイルの終りの次のレコードから、指定されたレコードの直前までのレコードを128バイトがゼロのレコードで埋め、その後、指定されたレコードに書き込みを行う。この点を除けばファンクション22H（ランダムな書き込み）と同じである。

●ランダムブロック書き込み

ファンクション: 26H

設定: DEレジスタ←オープンされたFCBの先頭アドレス

FCBのレコードサイズ←書き込むレコードサイズ

FCBのランダムレコード←書き込みを開始するレコード

HL←書き込むレコード数

DTA以降のメモリ領域←書き込むデータ

戻り値: 書き込みが成功すれば00Hを、失敗すれば01HをAレジスタにセットする。

書き込みが終わると、ランダムレコード・フィールドの値が自動的に更新されて、最後に書き込んだレコードの次のレコードを指す。ひとつのレコードの大きさはFCBのレコードサイズ・フィールドによって1バイトから65535バイトまで任意に設定できる。

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

●ランダムブロック読み出し

ファンクション： 27H

設定： DEレジスタ←オープンされたFCBの先頭アドレス
FCBのレコードサイズ←読み出すレコードサイズ
FCBのランダムレコード←読み出しを開始するレコード
HL←読み出すレコード数

戻り値： データの読み出しに成功すれば00Hを、失敗すれば01HをAレジスタにセットする。さらに、HLレジスタに、実際に読み込んだレコードの個数をセットする。

読み込みが終わると、ランダムレコード・フィールドの値が自動的に更新される。このシステムコール実行後、HLレジスタには実際に読み込んだレコードの総数がセットされる。つまり、指定した数のレコードを読み込み終わる前にファイルの終りに達してしまった場合には、それまでに読み込んだ実際のレコード数がHLレジスタに入ることになる。

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

4. 環境の設定と呼び出し

●日付の獲得

ファンクション: 2AH

設定: なし

戻り値: HLレジスタ←年 (1980~2079)
Dレジスタ←月 (1~12)
Eレジスタ←日 (1~31)
Aレジスタ←曜日 (0=日曜、1=月曜、…、6=土曜)

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

●日付の設定

ファンクション: 2BH

設定: HLレジスタ←年 (1980~2079)
Dレジスタ←月 (1~12)
Eレジスタ←日 (1~31)

戻り値: 設定に成功したかどうかをAレジスタにセットする。成功なら00H
が、失敗なら0FFHがセットされる。

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

●時刻の獲得

ファンクション: 2CH

設定: なし

戻り値: Hレジスタ←時
Lレジスタ←分
Dレジスタ←秒
Eレジスタ←1/100秒

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

●時刻の設定

ファンクション: 2DH

設定: Hレジスタ←時
Lレジスタ←分
Dレジスタ←秒
Eレジスタ←1/100秒

戻り値: 設定に成功したかどうかをAレジスタにセットする。成功なら00H
が、失敗ならFFHがセットされる。

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

●ベリファイ・フラグの設定

ファンクション： 2EH

設定： ベリファイ・フラグをリセットする時には、Eレジスタ←00H

ベリファイ・フラグをセットする時には、Eレジスタ←01H

戻り値： なし

ベリファイ・フラグをセットすると、以降のディスクに対する書き込みが、ベリファイ付きで行われるようになる。つまり、書き込んだあとでその内容をディスクから読みだして書き込むべき内容と比較して等しいかどうかをチェックする。

このシステムコールはMSX-DOS独自のもので、CP/Mとの互換性はない。

3 章 ディスクファイルの構造

システムコールを使用すれば、手軽に、しかもキメ細かくファイルを取り扱うことができます。ユーザーは、ディスク上でファイルがどのような方法で管理され、またどのような形式で記録されているか、といった情報を詳しく知る必要はありません。しかし、場合によっては、ディスクの管理情報を取り出したり、それをもとにファイルとは無関係にディスクを直接アクセスしたりする手段が必要になることがあります。

MSX-DOSでは、このような目的のために、ディスクの管理情報を得たり、“論理セクタ”を直接アクセスしたりするシステムコールを用意しています。

3. 1 ディスクファイルの構造

セクタを直接アクセスする場合には、どのセクタにどのような情報が書き込まれているか、という基本知識が必要になります。

● 論理セクタ

MSX-DOSでは、3.5インチフロッピーでもハードディスクでも、あるいはその他のドライブでも基本的にアクセスが可能です。それぞれのドライブや、あるいはメディアの種類によりセクタの大きさ、トラック毎のセクタ数、記録面の数などは違っていますが、これを統一的に管理するため、MSX-DOSでは、ディスク上の物理的な境界にとらわれず、すべてのセクタに、連続した通し番号を付けて、その番号でセクタを管理するという方法をとっています。これを“論理セクタ”と呼びます。“論理セクタ”（以下単に“セクタ”とも呼ぶ）の番号は、0からそのディスクの総セクタ数-1（ディスクの種類によって異なります）までの一連の番号によって指定されます。

MSX-DOSでは、ディスクの中のセクタを、表3.1に示す4つの領域に分けています。最初の3つの領域にはデータを管理するための情報が書き込まれ、ファイル・データは“データ領域”の部分に書き込まれています。これらの領域の位置関係は図3.1のとおりです。ブートセクタはかならずセクタ0に存在しますが、他の領域の開始セクタの位置などはメディアによって異なっています。ただし、そういった情報はブートセクタを読み出せばすべて得られるようになっています。

ブートセクタ	ディスク固有の情報とMSX-DOSの起動プログラム
FAT	ディスク上のファイルの位置情報
ディレクトリ	ディスク上のファイルの管理情報
データ領域	実際のファイル・データ

表 3.1 ディスクの領域

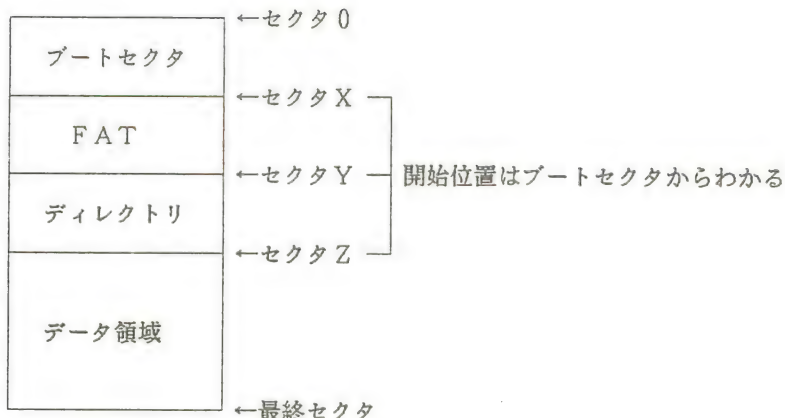


図 3.1 ディスク上の領域の位置関係

● クラスタ

ディスクとの入出力に関しては、前述のとおりセクタが基本単位です。ただし、ファイルに対してディスク上のセクタを割り当てる場合にはセクタ単位ではなく、複数のセクタから成る“クラスタ”という単位が使われ、それぞれのファイルには、そのファイルサイズに応じて必要な数のクラスタが割り当てられます。1クラスタよりも大きなサイズのファイルの場合、データは複数のクラスタにまたがって記録されます。1クラスタ未満の部分については、例えばそれが1バイトであっても1クラスタ分のデータ領域が割り当てられます。クラスタは論理セクタと同様に連続した番号で指定されていますが、FATの項で述べる理由で、2から始まる通し番号になっており、データ領域の先頭がクラスタ#2の位置に相当します。

● ブートセクタとDPB（ドライブ・パラメータ・ブロック）

MSX-DOSでは、接続されている個々のドライブごとに“DPB”という領域がメモリ上のワークエリアに設けられ、各ドライブに固有の情報が記録されます。MSX-DOSはどのようなタイプのディスクドライブにも対応可能となっていますが、それはこのDPBを参照して個々のドライブに対応した処理を行うことによって、メディア間の差異が吸収できるからに他なりません。

DPBに書き込まれる情報は、ディスク上のブートセクタに存在しているものであり、それがMSX-DOSの起動時、あるいはメディアが交換される毎に変更されます。ただし、ブートセクタとDPBとではその内容は図3.2と図3.3に示すように、異なった形式で記録されています。

0B		セクタサイズ (バイト単位)
0D		クラスタサイズ (セクタ単位)
0E		予約セクタ数 (F A T領域の先頭セクタ)
10		F A Tのコピー数
11		ディレクトリ・エントリ数 (作成可能なファイルの数)
13		総セクタ数
15		メディア I D
16		F A Tサイズ (セクタ単位)
18		トラックあたりのセクタ数
1A		ディスクの面数
1C		隠されたセクタ数

図 3.2 ブートセクタの情報

+0		ドライブ番号
+1		メディア I D
+2		セクタサイズ
+4		ディレクトリ・マスク
+5		ディレクトリ・シフト
+6		クラスタ・マスク (クラスタサイズ-1)
+7		クラスタ・シフト
+8		F A T領域の先頭セクタ
+10		F A Tのコピー数
+11		ディレクトリエントリ数
+12		データ領域の先頭セクタ
+14		最終クラスタ番号 (総クラスタ数+1)
+16		F A Tサイズ (セクタ単位)
+17		ディレクトリ領域の先頭セクタ
+19		F A Tバッファのアドレス (システム・メモリ)

図 3.3 D P B の構造

● F A T (ファイル・アロケーション・テーブル)

1 クラスタよりも大きなサイズのファイルは、複数のクラスタにまたがって記録されますが、その時連続した番号のクラスタが使用されるとは限りません。ファイルの作成／削除を何度も繰り返した後になると、解放されたクラスタがディスク上のあちこちに散在するようになりますが、この状態でサイズの大きなファイルを作成すると、データは飛び飛びのクラスタに分散して置かれます。そこで、何番目のクラスタは何番目のクラスタに続いている、というリンク情報を覚えておく場所が必要になります。それがF A Tの役割です。また、未使用クラスタの位置や、不良クラスタが発見された場合、以後そこをアクセスしないように記録する目的にもF A Tが利用されます。F A Tに記録されるこのようなクラスタのリンク情報や不良クラスタ情報は、ディスクファイルを管理するうえで不可欠なものであり、一部でも破損してしまうとディスク全体が使用できなくなる恐れがあります。そのためF A Tは常に複数個のコピーが用意され、万々に備えています。

F A Tの例を図3.4に示します。先頭の1バイトは“F A T I D”と呼ばれ、ディスクのメディアタイプを示す値、次の2バイトはダミー値のF F Hが入ります。そしてその次から、1クラスタにつき12ビットというフォーマットで実際のリンク情報を記録しているF A Tエントリが並んでいます。0番と1番に相当する3バイトが“F A T I D”に使われていますので、ファイルのデータに対応するF A Tエントリは2番から始まります。F A Tエントリの番号は、それに対応するクラスタの番号でもあります。F A Tエントリに記録された12ビットのリンク情報は、図3.5のように並んでいます。リンク情報は、次に続くクラスタ番号を示す値です。もしF F Hとなっている場合は、そのクラスタでファイルが終了したことを意味します。図3.4の例では、クラスタ#2→クラスタ#3→クラスタ#4、という3クラスタ分の大きさのファイルと、クラスタ#5→クラスタ#6、の2クラスタ分のファイルが存在していることがわかります（クラスタが番号の小さい順にリンクしているのは図を見やすくするために、実際には番号順であるとは限りません）。

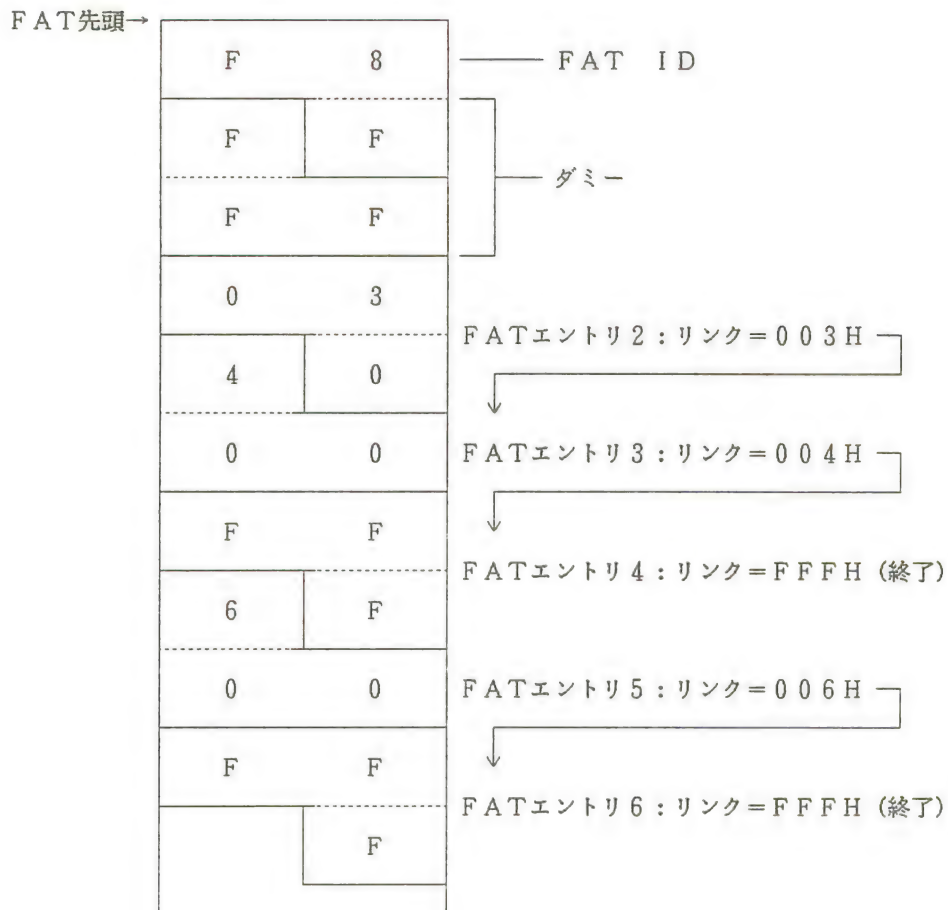


図 3.4 FAT の実例

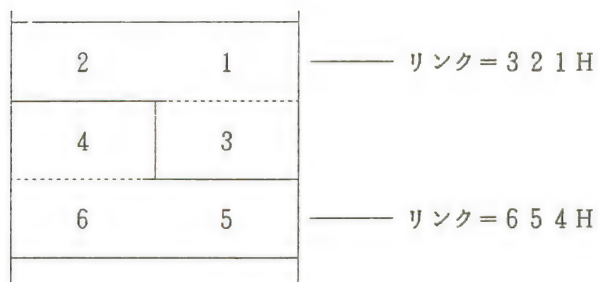


図 3.5 FAT の読み方

● ディレクトリ

FATは、データの位置関係などを表すものであり、ファイル自体に関する情報は含んでいません。したがって、そのファイルの名前やそれに付随する情報を知るには、FATとは別の情報源が必要です。これが“ディレクトリ”です。ディレクトリはディスク上のディレクトリ領域に記録されていて、図3.6に示すように32バイトごとにディレクトリ・エントリ（ディレクトリの格納場所）が並んでいます。ファイルの作成を行うと、使われていないディレクトリ・エントリの中で、いちばん番号が小さいところに目的のファイルのディレクトリが作られます。ファイルが削除されると、該当するディレクトリ・エントリの最初の1バイトにE5Hが書き込まれ、そのディレクトリ・エントリが空いたことを示します。ディレクトリ・エントリがすべて使用されてしまうと、データ領域がいくら残っていても新しいファイルを作ることはできません。

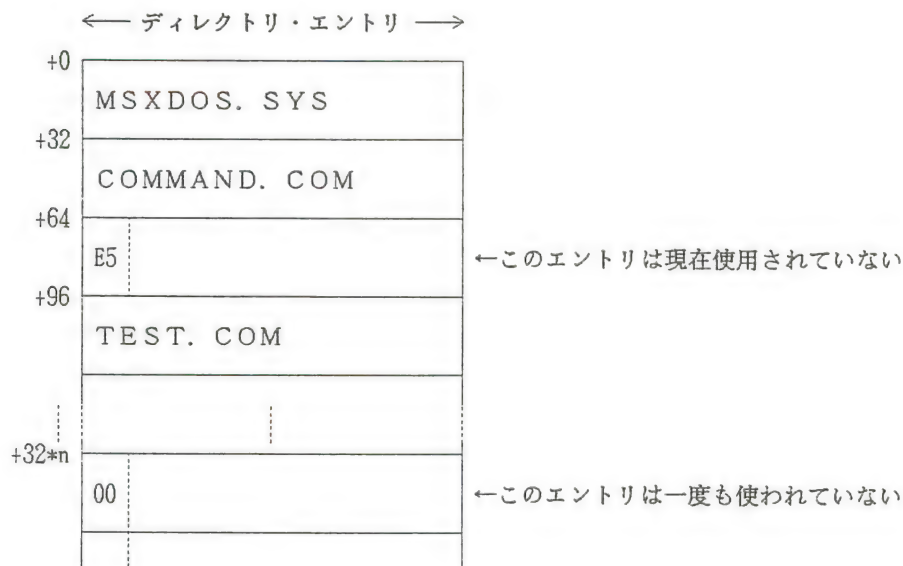


図 3.6 ディレクトリ領域の構造

ディレクトリ・エントリは図3.7のような構造を持っており、それぞれファイル名、ファイル属性、作成／更新の日時、ファイルの先頭クラスタ番号、ファイルサイズのを記録しています。ファイル属性はファイルに各種の属性（図3.8参照）を与えるものですが、MSX-DOSでは属性を持ったファイルを作ることはできません。ただし、既存のファイルに不可視／システム／ボリューム／ディレクトリのいずれかの属性が与えられていると、そのファイルは、システムコールではアクセスできないようになります。読み出し専用（書き込み禁止）属性は無視されます。日付と時刻は、図3.9と図3.10に示すように、それぞれ2バイトの領域を3つのビットフィールドに分割して記録しています。“年”は7ビットに0～99の値を設定することで、西暦1980年～2079年を表します。また“秒”のビットフィールドは5ビットで、時間の分解能は2秒となっています。

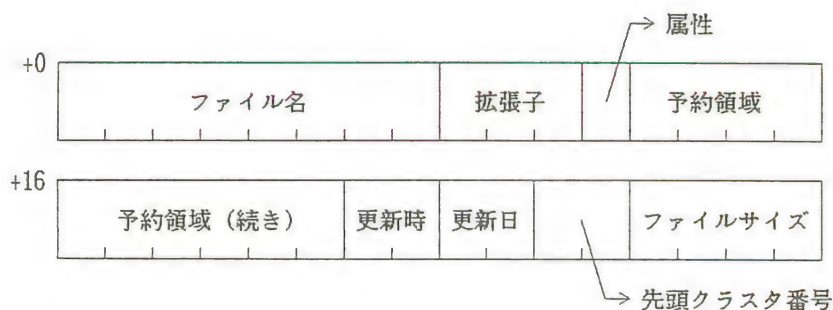


図 3.7 ディレクトリの構造

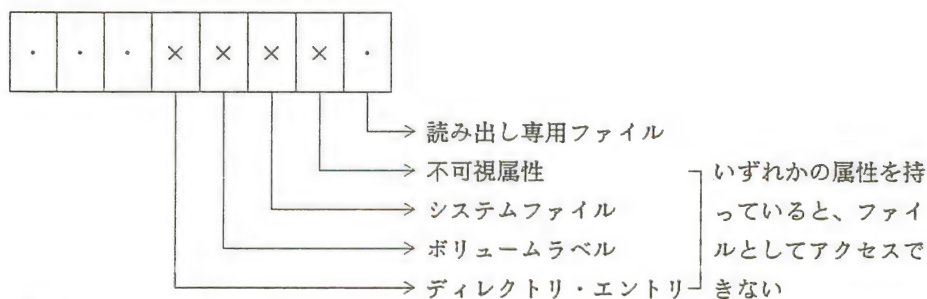


図 3.8 ファイル属性

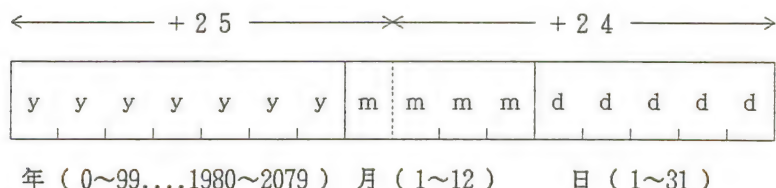


図 3.9 日付を表すビットフィールド

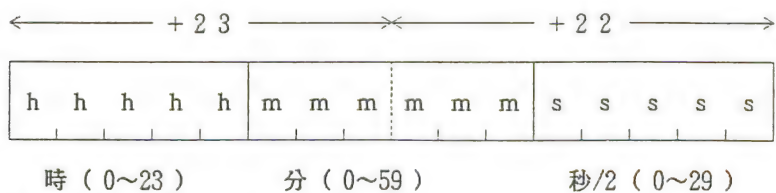


図 3.10 時刻を表すビットフィールド

3. 2 ディスク・アクセスのためのシステム・コール

本節では、ディスクの管理情報を得たり、論理セクタを使用してディスクを直接アクセスするためのシステムコールを説明します。トラックあたりのセクタの違いなどはMSX-DOSにまかせ、ユーザーは論理セクタだけですべてのディスクをアクセスすることができます。

●ディスク情報の獲得

ファンクション: 1BH

設定: Eレジスタ←目的のディスクが入っているドライブ番号

戻り値: Aレジスタ←1クラスタあたりの論理セクタ数
(指定ドライブがオンラインでない場合、FFH)
BCレジスタ←セクタのサイズ (バイト単位)
DEレジスタ←クラスタの総数
HLレジスタ←未使用クラスタの総数
IXレジスタ←DPBの先頭アドレス
IYレジスタ←FATバッファの先頭アドレス

指定ドライブのディスクの情報を得るシステムコールである。ドライブ番号に0を指定すると、デフォルト・ドライブの指定になる。それ以外は、Aドライブなら1、Bドライブなら2、……を指定する。

このファンクションはMSX-DOS独自のもので、CP/Mとの互換性はない。

●論理セクタの読み出し

ファンクション: 2FH

設定: DEレジスタ←読み出す論理セクタの番号 (複数の場合はその先頭の論理セクタ番号)

Hレジスタ←読み出す論理セクタの個数

Lレジスタ←読み出すディスクのドライブ番号
(A=0、B=1、… H=7)

戻り値: DTA以降の〔論理セクタサイズ×論理セクタの個数〕バイト
←読み込んだ内容

指定ドライブの指定論理セクタから、指定された個数の論理セクタを読み出し、その内容をDTA以降のメモリに格納する。もし、不適当な場所にDTAが設定されていると、そのメモリ領域が、読みだしたディスクの内容によって破壊されることになる。

このファンクションはMSX-DOS独自のもので、CP/Mとの互換性はない。

● 論理セクタの書き込み

ファンクション: 30H

設定: DEレジスタ←書き込む論理セクタの番号(複数の場合はその先頭の論理セクタ番号)

Hレジスタ←書き込む論理セクタの個数

Lレジスタ←書き込むディスクのドライブ番号

(A=0、B=1、… H=7)

DTA以降の〔論理セクタサイズ×論理セクタの個数〕バイト

←書き込むデータ

戻り値: なし

このファンクションはMSX-DOS独自のもので、CP/Mとの互換性はない。

● クラスタからセクタへの換算

FATやディレクトリでは、ディスク上のデータの位置はクラスタ単位で表わされています。クラスタで示されたこれらのデータをシステムコールでアクセスするためには、あるクラスタが何番のセクタに対応しているか、という関係を求めなければなりません。これは、データ領域がクラスタ#2から開始しているという事実を元に、以下のように計算することができます。

1. 与えられたクラスタ番号をCとする。
2. データ領域の開始セクタを調べ、これをS0とする。
3. 1クラスタが何セクタに相当するか調べ、これをnとする。
4. 求めるセクタ番号Sは $S = S0 + (C - 2) * n$ の計算で得られる。

索引I

ア行	
ウォームスタート	3 13
カ行	
外部コマンド	1 2 3 4 10
カレントブロック	6 7 8 17 19 20 22
カレントレコード	7 8 17 19 20 22
クラスタ	6 7 27 28 29 30 33 34 35
コンソール	4 12 13 14 15 16
サ行	
シーケンシャルアクセス	7 8
システムコール	8 10 11 12 13 15 16 17 18 20 22 23 24 25 26 32 34 35
システムスクラッチ・エリア	1 2 3
セクタ	12 17 26 27 28 29 34 35
タ行	
ディレクトリ	5 6 7 17 18 20 26 27 28 29 32 33 35
デフォルトドライブ	7 17 20
ハ行	
ファイルのオープン	5 12 17
ファイルのクローズ	5 12 17
ファンクション	3 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 34 35
ブート・セクタ	26 27 28
ラ行	
ランダム・アクセス	7 8 21 22
ランダム・ブロック・アクセス	6 7 8 9 22 23
ワ行	
ワイルドカード	18 19 20
B	
BASIC	13
BDOS	10
BIOS	3
C	
CP/M	8 9 13 15 23 24 25 34 35
D	
DTA	8 18 19 21 22 34 35
DPB	27 29 34

26	27	28	29	30	31	32	34	35	
1	5	6	8	17	18	19	20	21	22
23									

3 4' 32

10

3 4

MSX-DOS
SCREEN EDITOR

M. E. D.

OPERATION MANUAL

目 次

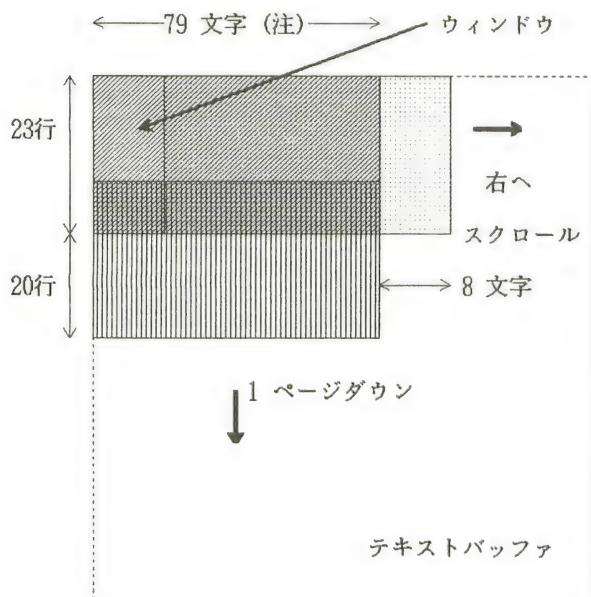
1	概 要	1
2	本書で用いる表記法	3
3	起動方法	5
4	コマンドの説明	7
4.1	コマンドの種類（形式別）	8
4.2	コマンド体系	11
4.3	モードの切換え	15
4.4	機能別コマンドの説明	16
4.4.1	カーソルの移動に関するコマンド	16
4.4.2	TABセットに関するコマンド	26
4.4.3	ウィンドウの移動に関するコマンド	27
4.4.4	削除に関するコマンド	30
4.4.5	挿入とコピーに関するコマンド	32
4.4.6	結合と分割に関するコマンド	34
4.4.7	サーチ（検索）、リプレース（置換）に関するコマンド	35
4.4.8	ブロック移動に関するコマンド	40
4.4.9	入出力に関するコマンド	46
4.4.10	ステータス表示に関するコマンド	48
4.4.11	エディタ（MED）の終了、初期化に関するコマンド	50
5	付 録	53
5.1	スクリーンエディタ（MED）のメッセージ	53



1 概要

MSX-DOSスクリーンエディタは、ソース・プログラムの作成や修正を行うために必要な数多くのコマンドを備えた本格的スクリーンエディタです。縦・横4方向のスクロールをサポートしています。編集のためのテキストバッファ容量は、約32Kbyteあります。また、一時記憶のためにヤンクバッファを設けてあります。

ディスプレイ上に画面として表示されている部分を、「ウィンドウ」といいますが、その動きについては以下の通りです。



- 1 79文字23行を画面表示します。
- 2 79文字を超える場合は、右へ8文字分スクロールします。
- 3 23行を超える場合は、1行ずつスクロールダウンします。
- 4 1ページアップ（ダウン）するというのは、20行分アップ（ダウン）することを意味します。

図 1

注) 1行の表示は、MSX₂の場合79文字、MSXの場合は39文字まで可能。

MSX₂においては、MSX-DOSスクリーンエディタを80文字モードでも使用することが出来ます。

その時は、MSX-DOSコマンドレベルで、以下のように入力してください。

A>MODE 80

カーソルの移動範囲は、文字のある所（空白を含む）に限られます。

このスクリーンエディタは、アスキー形式のファイル以外は編集できません。

使用可能な文字は、以下に示す通りです。

- ・ キャラクターコード表（16進コード）の20～FD

（但し、7F、A0は除きます）

- ・ 0CH (CTRL+L) 及びEOFマークとしての1AH (CTRL+Z)、
09H (TAB) 及び改行文字である0DH (CR)、0AH (LF) がその順
番に現れる場合。

メモリサイズが0になる場合は、入力不能になります。

ダイレクトコマンドのようにキー入力後直ちに実行される場合を除き、ESCキー
またはCTRL+Cによりコマンドの実行を中止することができます。

（ESCキーにより中止した場合は、コマンド入力待ち状態になります。）

2 本書で用いる表記法

本書ではコマンドやステートメントの解説に、次のような表記法を用いています。

[] 角形カッコ……このカッコの中身は、必要に応じて入力する項目を指定します。

< > 山形カッコ……このカッコの中身は、ユーザーが入力するデータを示します。

テキストが山形カッコで囲まれている場合には、そこに指定された項目を入力します。

(例) <ファイル名>

{ } 大カッコ……このカッコは、そこに並べられた項目のうち、必要なものを選んで入力することを表します。角形カッコで囲まれていない限り、必ず1つは入力しなければなりません。

… 省略記号……必要に応じて何度か繰り返して入力する項目を示します。

| 縦線……これは選択項目の区切りに使われます。また、フィルタとして使った場合には、パイプを意味します。画面上には|と表示されます。

大文字……その綴りどおりに入力しなければならないステートメントやコマンドを意味します。また、特殊文字も表します。

特にことわりがない限り、大文字・小文字の区別はありません。

カンマ (,) , コロン (:) , スラッシュ (/) , 等号 (=) などの記号は、表示されているとおりに、その位置に入力しなければなりません。

複数の指定を同時に行う場合、各指定項目の間はスペースで区切ります。

《例》

MED_ファイル名1_ファイル名2
| |
_____ スペースで区切る

RETURN --- RETURNキーを表します。

CTRL ----- CTRLキーを表します。

ESC ----- ESCキーを表します。

SELECT --- SELECTキーを表します。

INS ----- INSキーを表します。

TAB ----- TABキーを表します。

BS ----- BSキーを表します。

DEL ----- DELキーを表します。

SHIFT ----- SHIFTキーを表します。

HOME ----- HOMEキーを表します。

F1 ----- F1キーを表します。

F2 ----- F2キーを表します。

F3 ----- F3キーを表します。

F4 ----- F4キーを表します。

F5 ----- F5キーを表します。

F6 ----- F6キーを表します。

F7 ----- F7キーを表します。

F8 ----- F8キーを表します。

F9 ----- F9キーを表します。

F10 ----- F10キーを表します。

CTRL+Cのような表現は、コントロール文字、特殊文字の入力を示します。

これは、CTRLを押しながらCを押すという意味です。

3 起動方法

MSX-DOSスクリーンエディタは、以下のように入力することで起動されます。

```
MED [<ロードファイル名>] [<セーブファイル名>] RETURN
```

ファイル名を省略した場合は、図2に示す画面になります。この画面は、MED.MES というファイルを読み込んで表示しており、図2に示したものはその標準的な内容です。

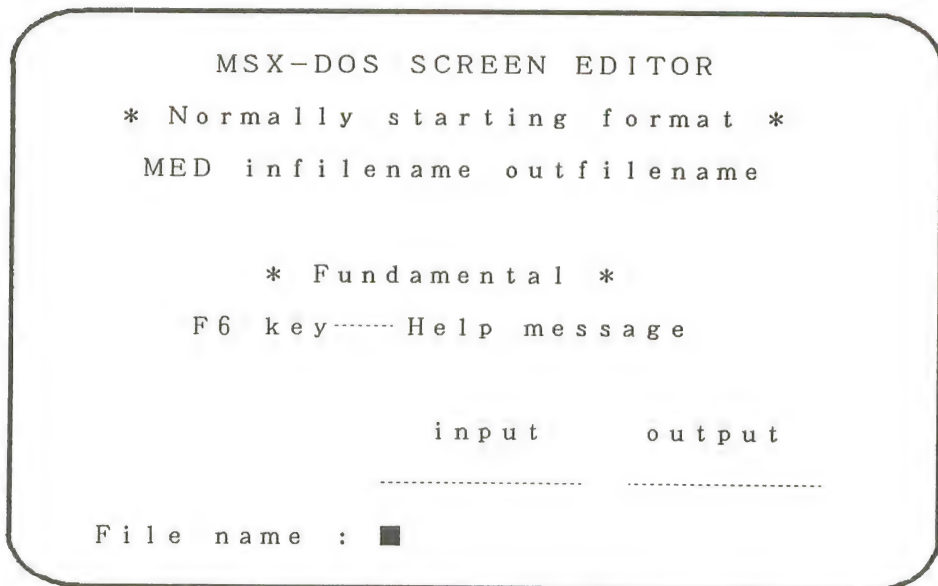


図2

ファイル名を入力してください。ファイル名を入れずに、RETURNキーを押すと、次ページの図3に示す画面になります。

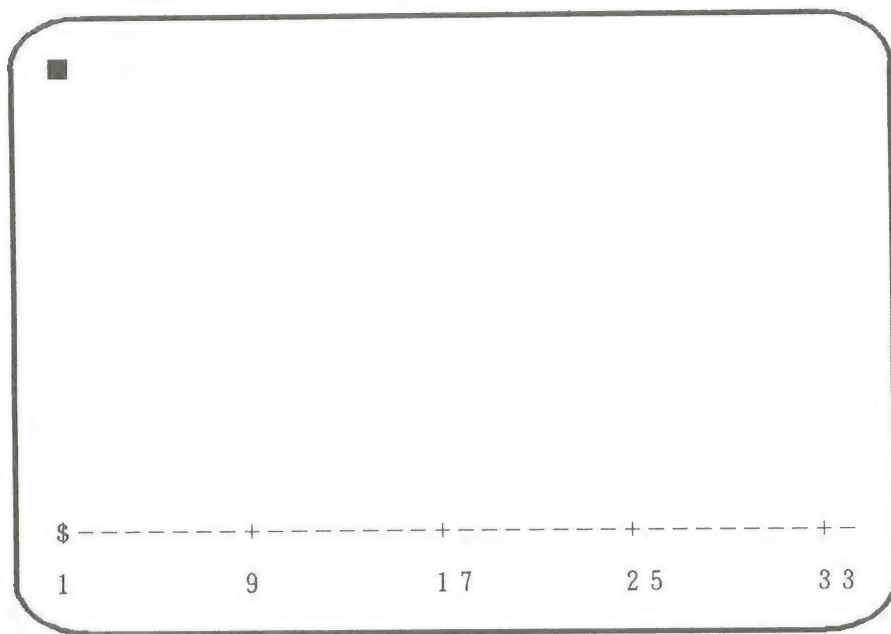
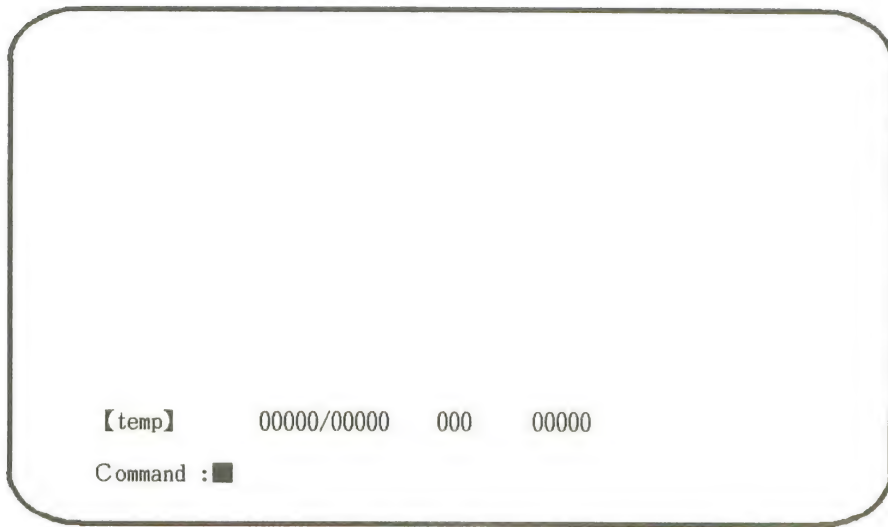


図 3

この状態でノーマルモードでの文字の入力ができ、ダイレクトコマンドが使えます。

スケール上の\$マークは、カーソルの現在の横位置を示します。

また、ESCキーを押すことにより、画面下のスケールが次ページの図4のような表示になります。



① ② ③ ④ ⑤

図 4

図4 における最下行のメッセージの説明は、次の通りです。

① 現在のファイル名

最初は "temp" という名前がつけられていますが、ファイルのリードやライトによって名前をかえることができます。これは、ファイルをライトする時のデフォルト名になります。

② 現在の行番号

現在カーソル行が何行目にあるかを表示します。先頭行は1行目になります。

③ テキストバッファ内の最下行の行番号

テキストバッファ内の最下行が何行目であるかを表示します。行数のカウントは1行から始まります。

④ カーソルの現在の横位置

カーソルが現在その行のn文字目にあることを表示します。(nは1～65535)
この数字は、ウィンドウ内の位置ではなく、行内の位置を示します。

⑤ エディットバッファの空き領域

エディットバッファの空き領域があと何バイトあるかを表示します。

4 コマンドの説明

4.1 コマンドの種類 (形式別)

MSX-DOSのスクリーンエディタで使用するコマンドを形式別に大別すると以下の通りです。

項番	種類	入力形態	機能概要
1	ダイレクト コマンド (直接実行 コマンド)	特殊キー コントロールキー	カーソル移動、ウィンドウ移動 挿入、削除、モード切換え 貼り込み 行の分割・結合・改行 e t c.
2	インダイレクト コマンド (間接実行 コマンド)	ESCキー + コマンド名	ファイルの読み込み、書き出し 画面出力・プリンタ出力 ディレクトリ表示 フロッピーディスク諸元の表示 ヘルプファイルの表示 他モジュールの起動 テキストバッファの初期化 ブロックの書き出し・付加 e t c.
3	ブロック移動 コマンド	SELECTキー + コマンド名	ブロックのコピー・消去 ブロックのバッファへの記憶
※	ファンクション キー	F α キー 〔 α は1～10の 整数〕	項番1・2・3のキーの中で使用頻 度の高いものを、1つの機能キーと して登録したもの。

表1 コマンドの種類 (形式別)

① ダイレクトコマンド

- ・特殊キー
- ・コントロールキー

キーボード上にあるキーを入力することにより、直接コマンドが起動します。

② インダイレクトコマンド

ESCキー+<コマンド名>

ESCキーを入力すると、ウィンドウの最下行に、下に示すメッセージを出力します。

command : コマンド名

コマンド名を入力すると、コマンドが起動します。

インダイレクトコマンドは、コマンド入力後RETURNキーを押すことにより実行します。

③ ブロック移動コマンド

SELECTキー+<コマンド名>

SELECTキーを入力すると、ウィンドウの最下行に下に示すメッセージが順次出力されます。

① Please set start point

② Please set end point

③ COPY DELETE BUFFER — SPACE —

または

WRITE APPEND — SPACE —

(SPACEキーによって、表示されるメッセージを切り換えられます。)

次にコマンド名を入力すると、コマンドが起動します。

④ コマンドの実行中止

ダイレクトコマンドのようにコマンド入力後直ちに実行される場合を除き、
ESCキー又はCTRL + Cによりコマンドの実行を中止することができます。

ESCキーにより実行中止した場合は、そのままインダイレクトコマンド入力が
可能な状態になります。

4.2 コマンド体系

エディット

ページ

ファンクションキー

F 1	テキストの最初の行に移動 (ESC + 0)	(カーソル移動)	2 3
F 2	テキストの末尾の行に移動 (ESC + \$)	(カーソル移動)	2 4
F 3	文字列のサーチ	(ESC + S) (検索)	3 5
F 4	リプレース	(ESC + R) (置換)	3 7
F 5	再実行	(CTRL + Q) (検索, 置換)	3 9
F 6	コマンドの一覧表を表示	(ESC + H) (ステータス表示)	4 9
F 7			
F 8			
F 9			
F 10			

ダイレクトコマンド

→	カーソルを右に移動	(カーソル移動)	1 8
←	カーソルを左に移動	(カーソル移動)	1 8
↑	カーソルを上に移動	(カーソル移動)	1 8
↓	カーソルを下に移動	(カーソル移動)	1 8
INS	モード切り換え	(モード)	1 5
RETURN	改行	(カーソル移動)	1 6 · 1 7
HOME	ウィンドウの左上隅に移動	(カーソル移動)	1 9
TAB	水平タブ	(8文字単位・固定)	2 6
BS	カーソル位置の左側1文字を削除	(削除)	3 0
DEL	カーソル位置の文字を削除	(削除)	3 0

(次ページへ)

CTRL

ページ

— R	モード切り換え	(モード)	1 5
— M	改行	(カーソル移動)	1 6・1 7
— K	ウィンドウの左上隅に移動	(カーソル移動)	1 9
— T	ウィンドウの左下隅に移動	(カーソル移動)	2 0
— F	1 単語分、右に移動	(カーソル移動)	2 1
— B	1 単語分、左に移動	(カーソル移動)	2 1
— C	行の先頭に移動	(カーソル移動)	2 2
— V	行の末尾に移動	(カーソル移動)	2 2
— I	水平タブ	(8 文字単位・固定)	2 6
— S	右スクロール	(ウィンドウ移動)	2 7
— A	左スクロール	(ウィンドウ移動)	2 7
— U	アップ	(ウィンドウ移動)	2 8
— D	ダウン	(ウィンドウ移動)	2 8
— W	ページアップ	(ウィンドウ移動)	2 9
— Z	ページダウン	(ウィンドウ移動)	2 9
— H	カーソル位置の左 1 文字を削除	(削除)	3 0
— Y	カーソル行を削除	(削除)	3 1
— E	カーソル位置以降を行末まで削除	(削除)	3 1
— O	カーソル行の上に、空行 1 行を挿入	(挿入)	3 2
— P	ヤンクバッファの内容を挿入	(挿入)	3 3
— J	カーソル行と次の行を結合	(結合)	3 4
— N	カーソル位置から右側を分割	(分割)	3 4
— Q	F 5 と同じ	(検索・置換)	3 9
— X	SELECT と同じ	(ブロック移動)	4 0

(次ページへ)

ESC			ページ
	インダイレクトコマンド		
0	テキストの最初の行に移動	(カーソル移動)	2 3
\$	テキストの最後の行に移動	(カーソル移動)	2 4
数字 (行番号)	指定された行に移動	(カーソル移動)	2 5
数字 (個数)	F3 — { ↑ ↓ RETURN }		
	指定された個数分実行	(検索)	3 5
	F4 指定された個数分実行	(置換)	3 7
	F5 指定された個数分再実行	(検索・置換)	3 9
READ	テキストをメモリに読み込む	(入出力)	4 6
DIR	ディレクトリの表示	(ステータス表示)	4 8
DSKF	フロッピーディスクの諸元を表示		
		(ステータス表示)	4 8
HELP	コマンドの一覧表を表示	(ステータス表示)	4 9
NEW	エディットバッファのクリア	(初期化)	5 0
QUIT	エディタの終了	(終了)	5 1
UPDATE	ファイルの更新	(終了)	5 2
SELECT	→次ページのSELECT参照		

(次のページへ)

- 《注意》
- ・READ, QUIT, HELP, UPDATEについては、それぞれ先頭の1文字のみの入力も可能です。
 - ・DIR, DSKF, NEWについては、すべての綴りを入力して下さい。

SELECT

ページ

ブロック移動コマンド

COPY	ブロックのコピー	(コピー)	40・42
DELETE	ブロックの消去	(削除)	40・43
BUFFER	ブロックをヤンクバッファに記憶		40・43
WRITE	テキストをディスクに書き込む (入出力)		40・44
APPEND	テキストをディスク上のファイルに付加する	(入出力)	40・45

《注意》 ・COPY, DELETE, BUFFER, WRITE, APPENDは
先頭の1文字のみ入力します。

4.3 モードの切換え

MSX-DOSのスクリーンエディタは、2つのモード（ノーマル・モードとインサート・モード）をもっています。同じコマンドでも、モードによって機能が異なる場合がありますので、注意してください。以下に、それぞれのモードについて説明します。

① モード切り換え

キーボード最上部にあるINSキーが、トグル・スイッチになっています。

また、コマンドCTRL+Rも同じ機能をもっています。

② 2つのモード

・ノーマル・モード（上書き）

カーソル形態がボックス型（■）の場合、ノーマル・モードです。

カーソル位置に入力文字が上書きされます。

・インサート・モード（挿入モード）

カーソル形態がライン型（—）の場合、インサート・モードです。

カーソル位置の左側に文字を挿入します。

MSX-DOSのスクリーンエディタでは、ノーマル・モードの状態で起動されますので、必要に応じてINSキー、あるいはCTRL+Rで、切り換えてください。

4.4 機能別コマンドの説明

4.4.1 カーソルの移動に関するコマンド

《入力形態》 RETURN | CTRL+M

《機能》 改行をします。

《説明》

① ノーマル・モードの場合

下図のように、カーソル行の次行の先頭にカーソルを移動します。

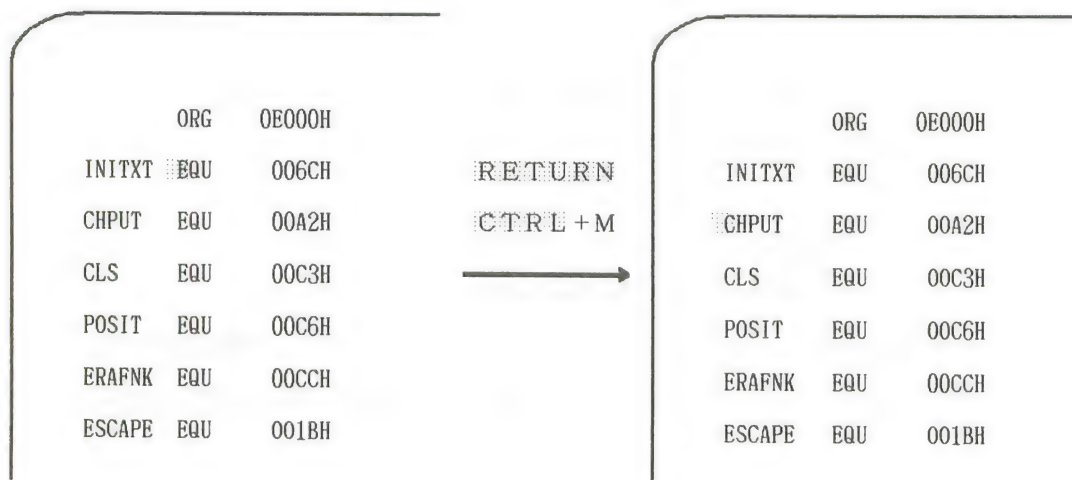


図 5

②インサート・モードの場合

下図のように、カーソル位置以降（カーソル位置を含む）を分割し、次の行に新しい行として挿入します。

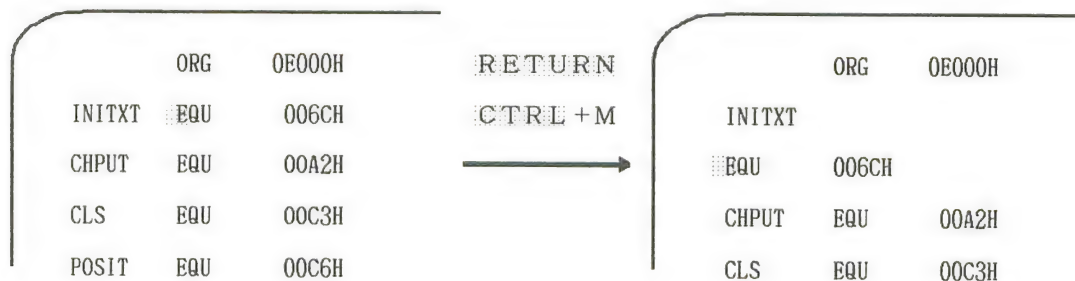


図 6

《入力形態》 → (右) | ← (左) | ↑ (上) | ↓ (下)

《機能》 カーソルを右左上下に移動します。

《説明》 下図のように、カーソル位置を移動します。

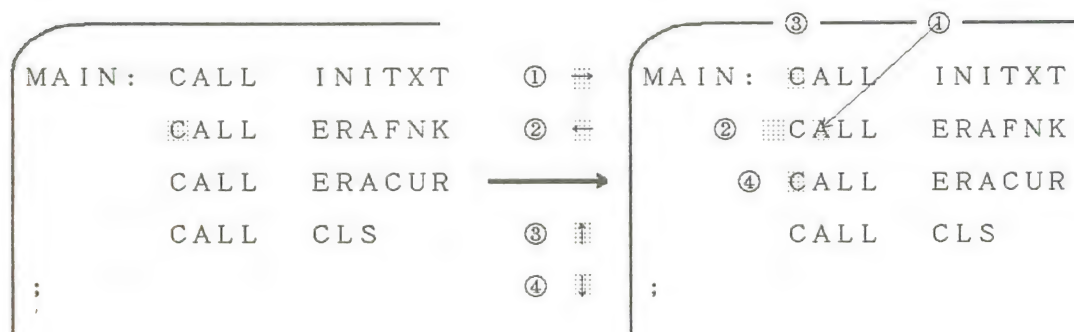


図 7

《注意》 カーソルの状態が下記の場合には、カーソルは移動しません。

- ① → …テキストの末尾
- ② ← …テキストの先頭
- ③ ↑ …テキストの開始行
- ④ ↓ …テキストの最終行

《入力形態》 HOME | CTRL+K

《機能》 カーソルをウィンドウの左上隅に移動します。

《説明》 下図のように、カーソル位置を移動します。

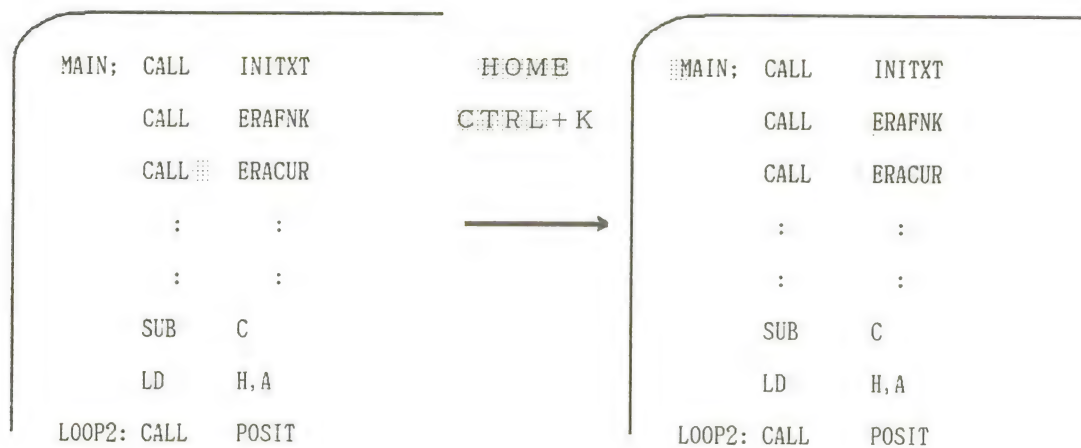


図 8

《入力形態》 CTRL + T

《機能》 カーソルをウィンドウの左下隅に移動します。

《説明》 下図のように、カーソル位置を移動します。

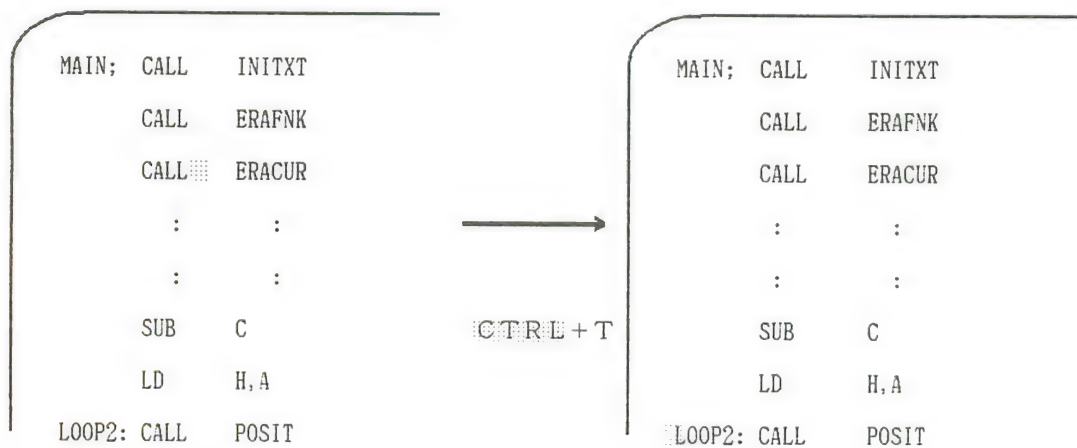


図 9

《入力形態》 `CTRL + F`

《機能》 カーソルを1単語分、右に移動します。つまり、次の単語の先頭にカーソルが移動します。

《説明》 下図のように、カーソル位置を移動します。

《注意》 カーソル位置以降に単語がない場合、カーソルは行の末尾に移動します。

《入力形態》 `CTRL + B`

《機能》 カーソルを1単語分、左に移動します。つまり、前の単語の先頭にカーソルが移動します。

《説明》 下図のように、カーソル位置を移動します。

《注意》 カーソルが単語内にある場合、カーソルはその単語の先頭に移動します。

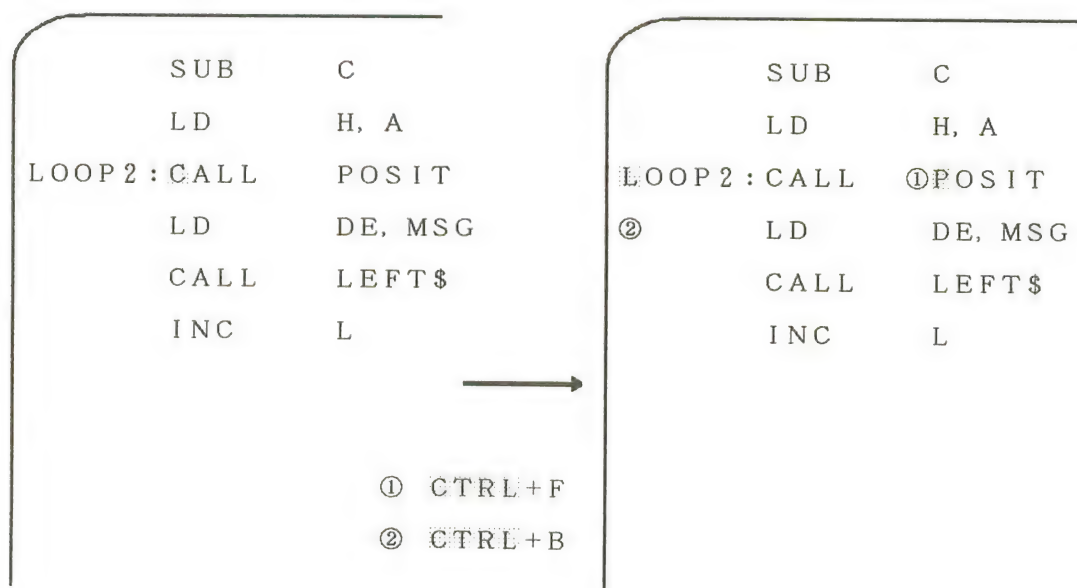


図10

《入力形態》 CTRL + C

《機能》 カーソルを行の先頭に移動します。

《説明》 下図のように、カーソルを移動します。

《入力形態》 CTRL + V

《機能》 カーソルを行の末尾に移動します。

《説明》 下図のように、カーソルを移動します。

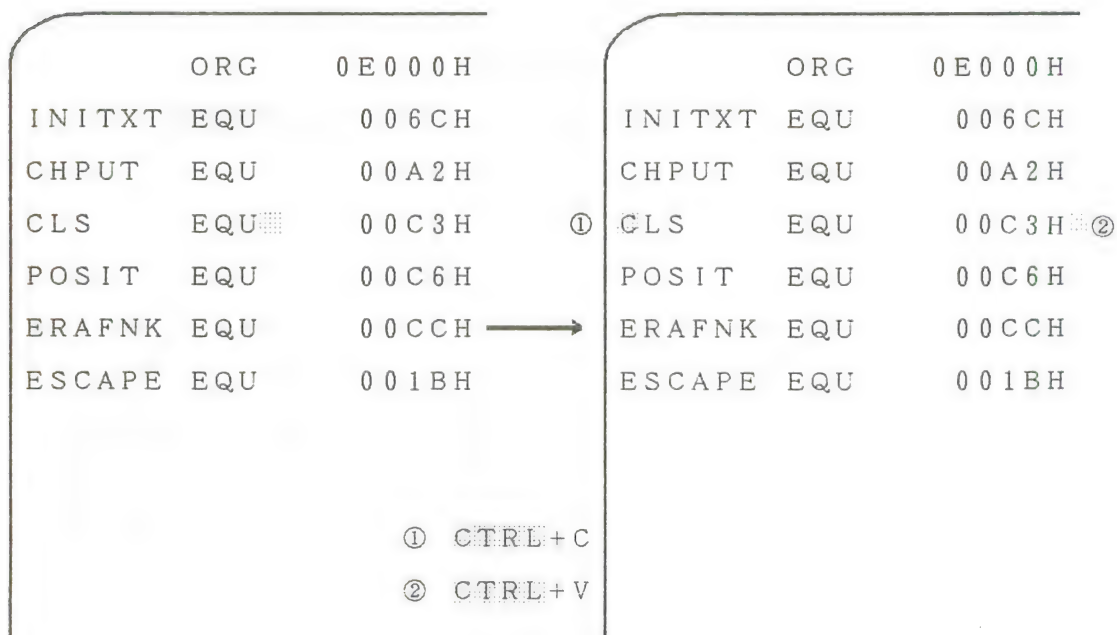


図 1 1

《入力形態》 F1 | ESC+0+RETURN

《機能》 カーソルをテキストの最初の行に移動します。

《説明》 下図のように、カーソルを移動します。

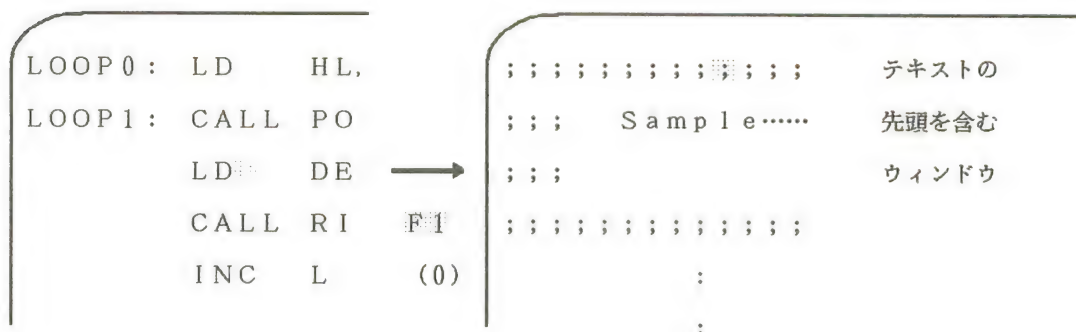


図12

《入力形態》 F2 | ESC+\$+RETURN

《機能》 カーソルをテキストの末尾に移動します。

《説明》 下図のように、カーソルを移動します。

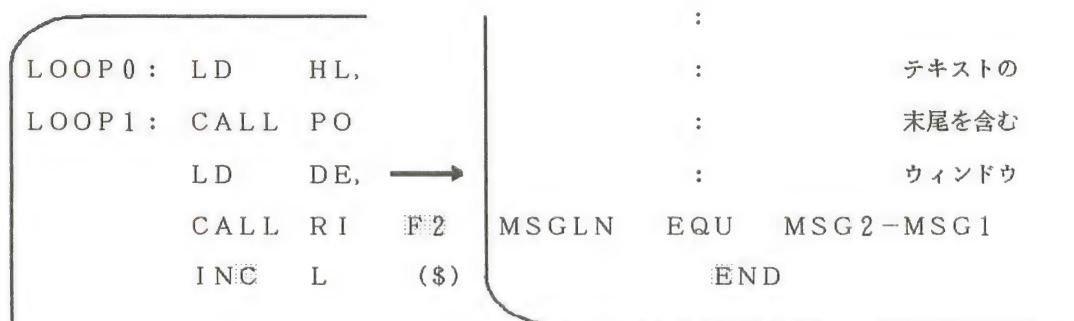
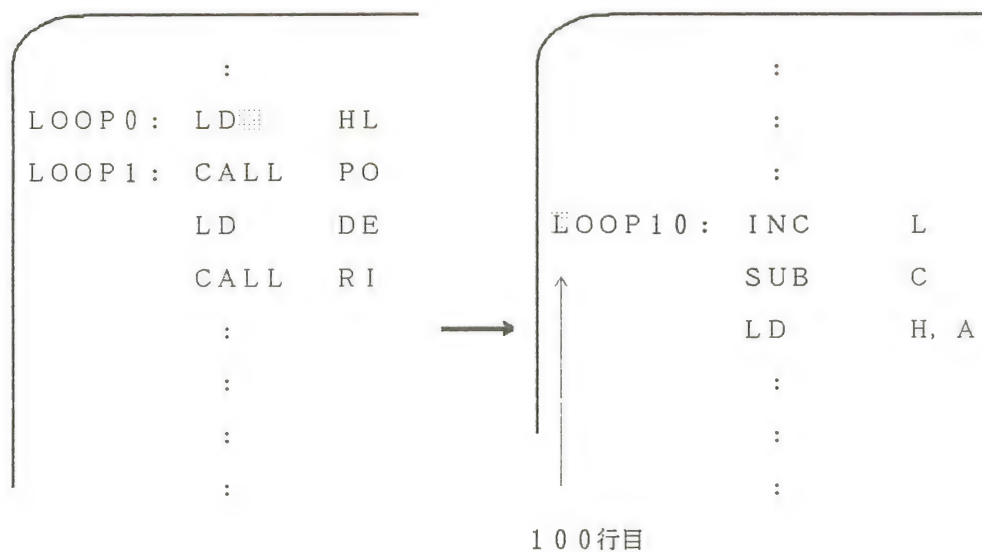


図13

《入力形態》 ESC+行番号+RETURN

《機能》 カーソルを指定された行に移動します。

《説明》 下図のように、カーソルを移動します。



ESC+100+RETURN

图 14

4.4.2 TABセットに関するコマンド

《入力形態》 **TAB** | **CTRL+I**

《機能》 8文字単位（固定）に水平タブを行います。

《説明》 ①ノーマル・モードの場合

- ・カーソル位置に文字がある時、その文字がタブに変わります。
 - ・カーソル位置に文字がない時、カーソル位置にタブが書き込まれます。
- いずれの場合にも、カーソルは次のタブ位置に移動します。

②インサート・モードの場合

カーソル位置の前に、タブが挿入され、カーソルは次のタブ位置に移動します。

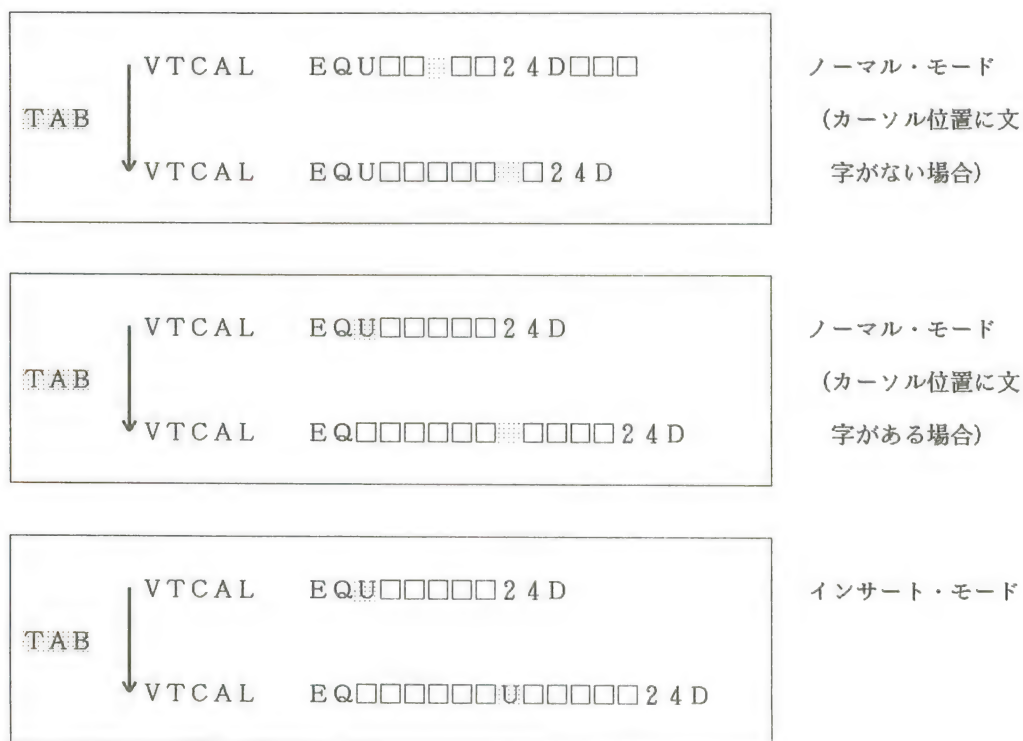


図 15

4.4.3 ウィンドウの移動に関するコマンド

《入力形態》 CTRL+S

《機能》 ウィンドウを8文字単位に、右へスクロールします。

《説明》 下図のように、ウィンドウが移動します。

《入力形態》 CTRL+A

《機能》 ウィンドウを8文字単位に、左へスクロールします。

《説明》 下図のように、ウィンドウが移動します。

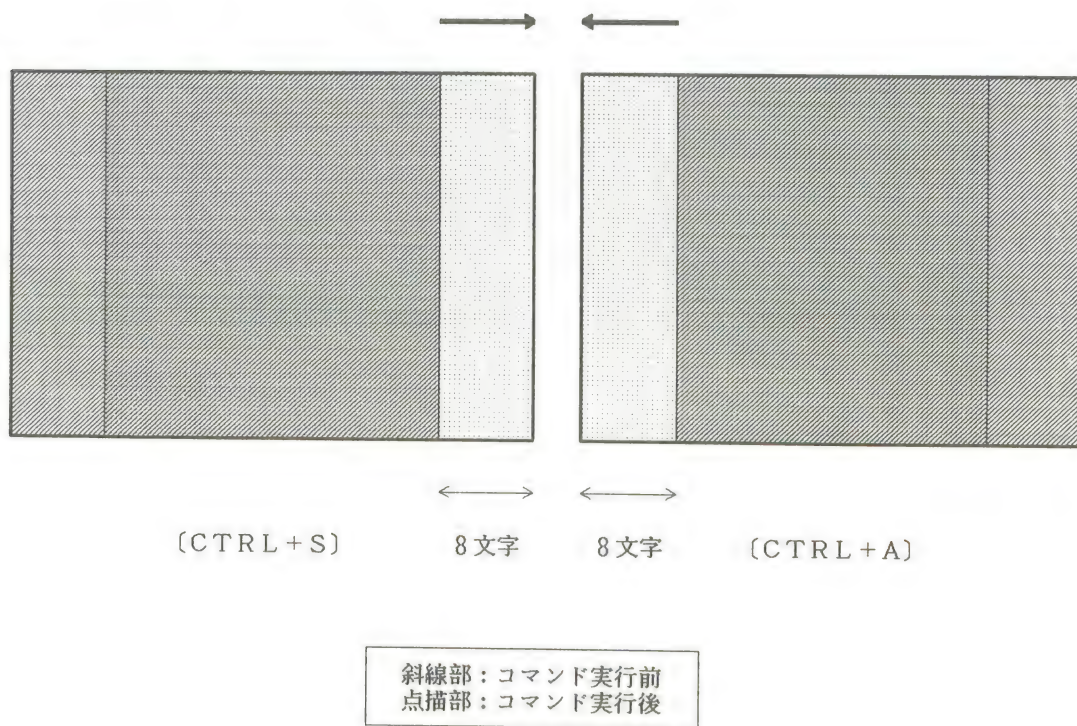


図 16

《入力形態》 **CTRL + U**

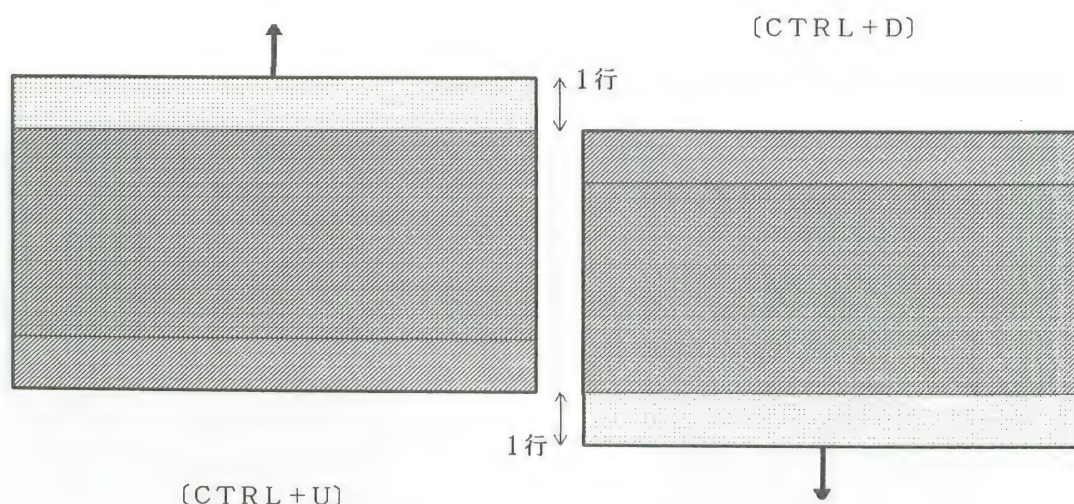
《機能》 ウィンドウを1行単位に、アップします。

《説明》 下図のように、ウィンドウが移動します。

《入力形態》 **CTRL + D**

《機能》 ウィンドウを1行単位に、ダウンします。

《説明》 下図のように、ウィンドウが移動します。



斜線部：コマンド実行前
点描部：コマンド実行後

図 17

《入力形態》 CTRL+W

《機能》 ウィンドウを1ページ（20行単位）分、ページアップします。

《説明》 下図のように、ウィンドウが移動します。

《入力形態》 CTRL+Z

《機能》 ウィンドウを1ページ（20行単位）分、ページダウンします。

《説明》 下図のように、ウィンドウが移動します。

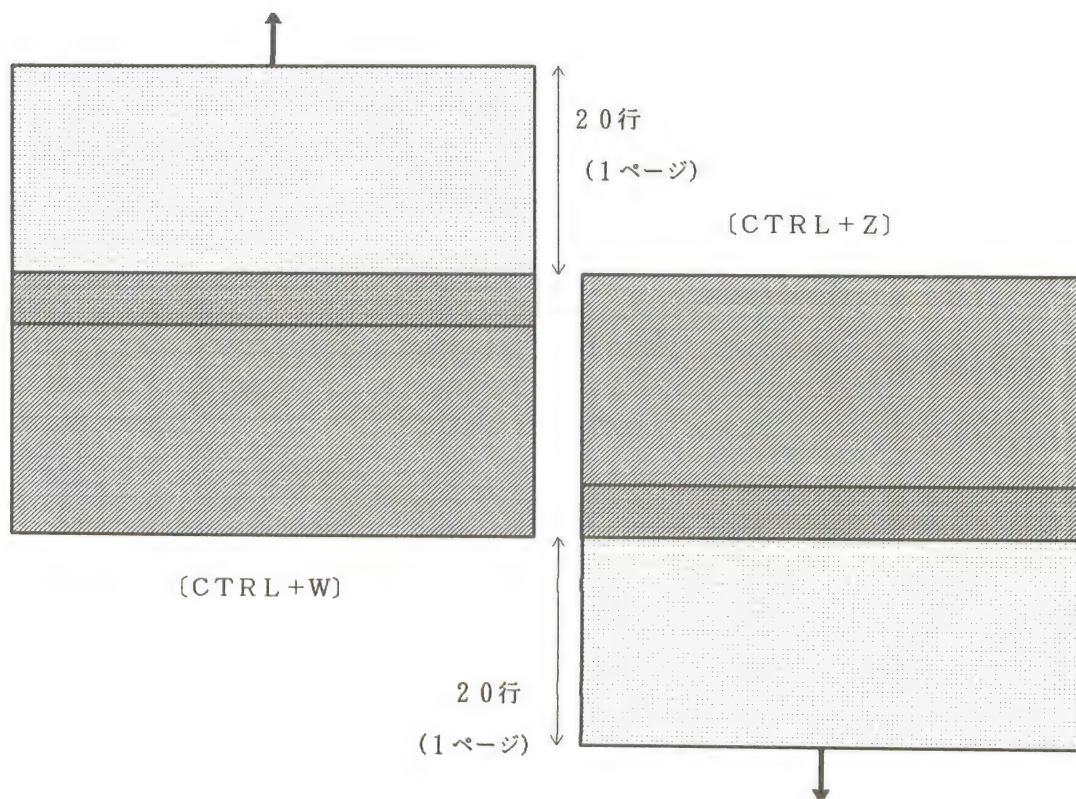


図 18

4.4.4 削除に関するコマンド

《入力形態》 BS | CTRL+H

《機能》 カーソル位置の左側1文字を削除します。

《説明》 下図のように、文字削除した後、左詰めになります。

《入力形態》 DEL

《機能》 カーソル位置の文字を削除します。

《説明》 下図のように、文字削除した後、左詰めになります。

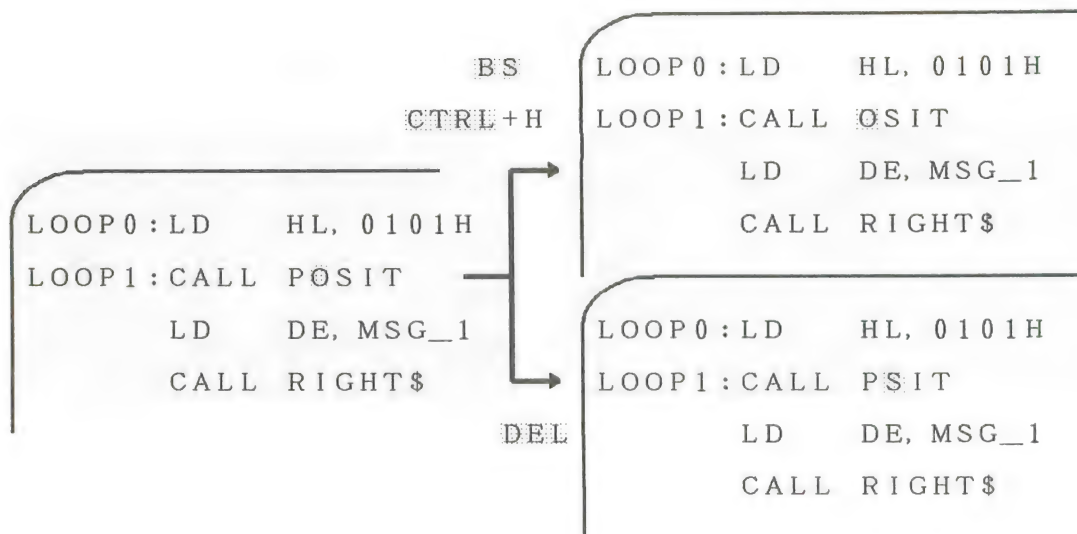


図 19

《入力形態》 CTRL+Y

《機能》 カーソル行を削除します。

《説明》 下図のように、行削除した後、次の行以降前詰めになり、カーソルは次の行の先頭に移動します。

《入力形態》 CTRL+E

《機能》 カーソル位置以降（カーソル位置を含む）を行末まで削除します。

《説明》 下図のように削除した後、カーソル位置は変わりません。

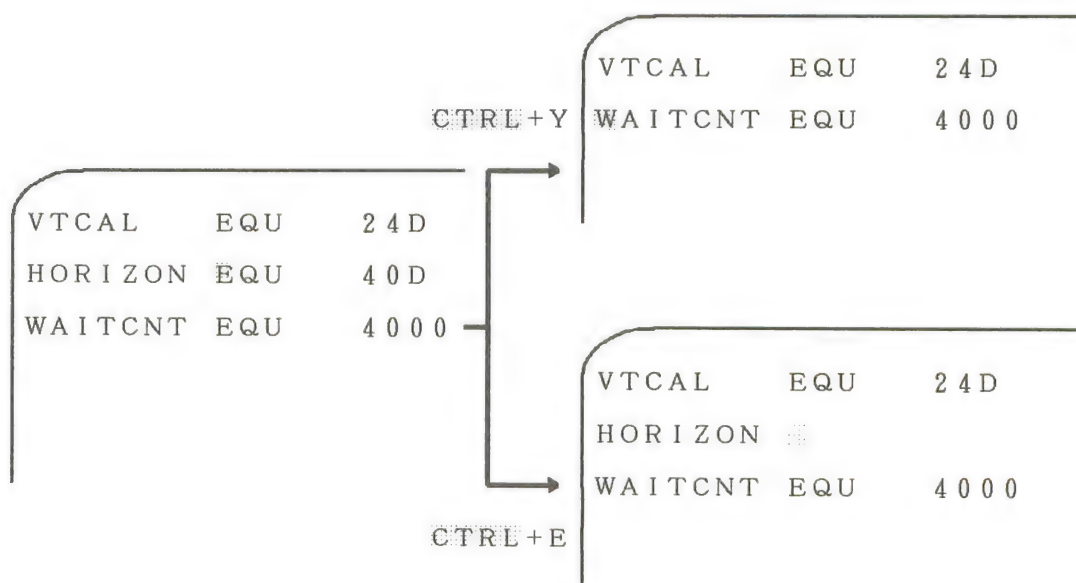


図 20

4.4.5 挿入とコピーに関するコマンド

《入力形態》 CTRL+O

《機能》 カーソル行の上に、空行1行を挿入します。

《説明》 下図のように、空行を挿入した後、カーソルは空行の先頭に移動します。

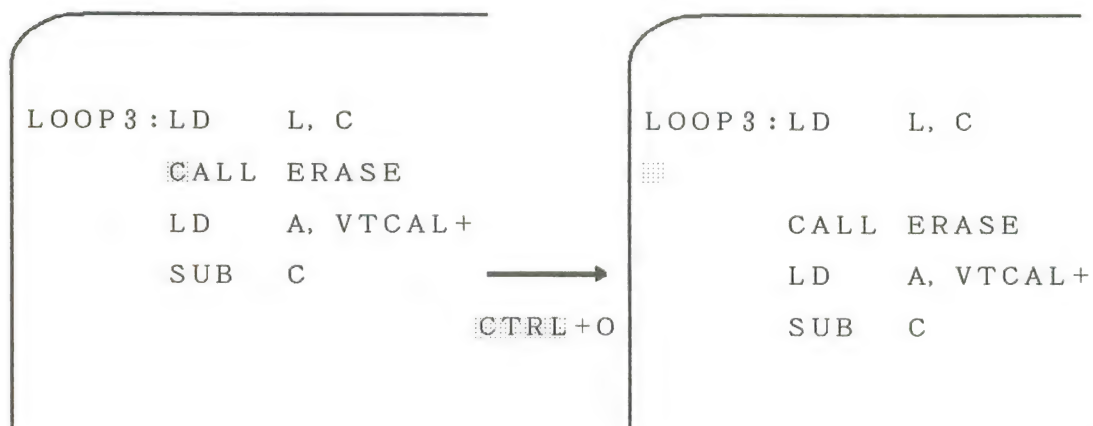


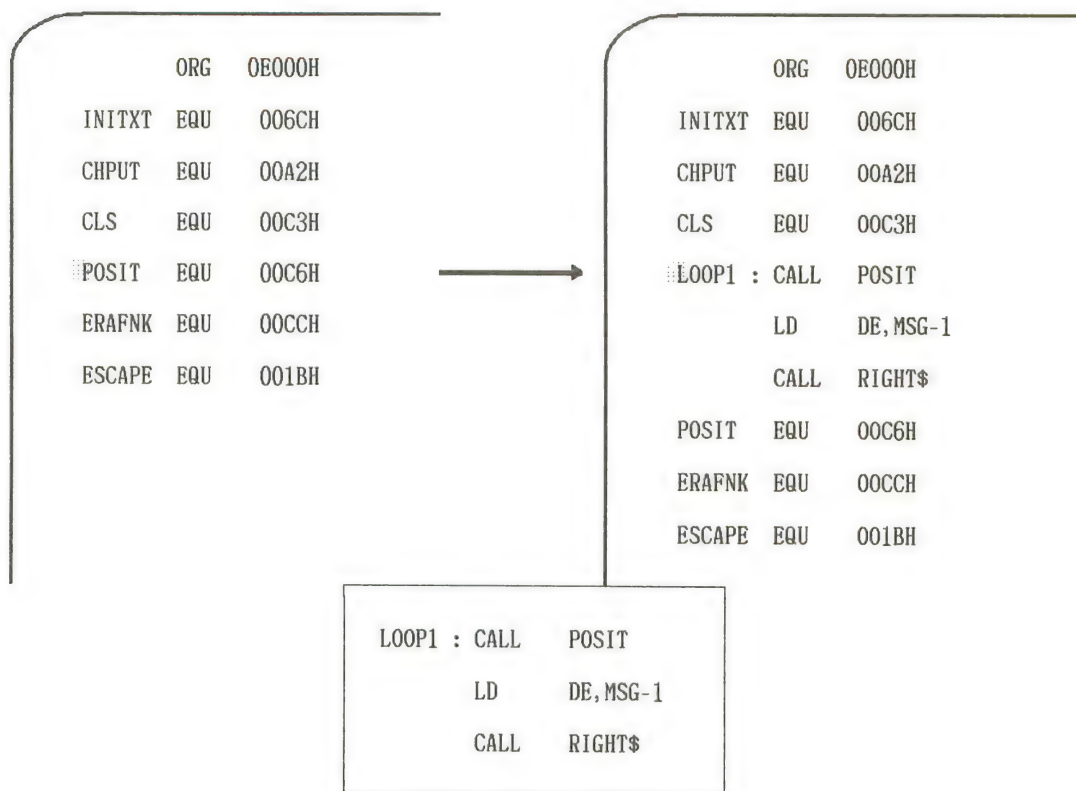
図 21

《入力形態》 CTRL + P

《機能》 カーソル位置以降に、ヤンクバッファの内容を挿入します。

《説明》 下図のように、カーソル位置以降に貼り込みが行われます。ヤンクバッファの内容は、クリアされませんので、何度でも貼り込みができます。

《注意》 貼り込みの結果、テキストバッファの容量が不足する場合、エラーとなり CTRL + P は実行されません。



ヤンクバッファの内容

図 2 2

4.4.6 結合と分割に関するコマンド

《入力形態》 **CTRL + J**

《機能》 カーソル行と次の行を結合して1つの行にします。

《説明》 下図のように行の結合が行われますが、カーソル位置は変わりません。

《注意》 カーソル行が、テキストの最終行である場合、**CTRL + J**は実行されません。

《入力形態》 **CTRL + N**

《機能》 カーソル位置から右側を分割し、2行にします。

《説明》 下図のように行の分割が行われますが、カーソル位置は変わりません。

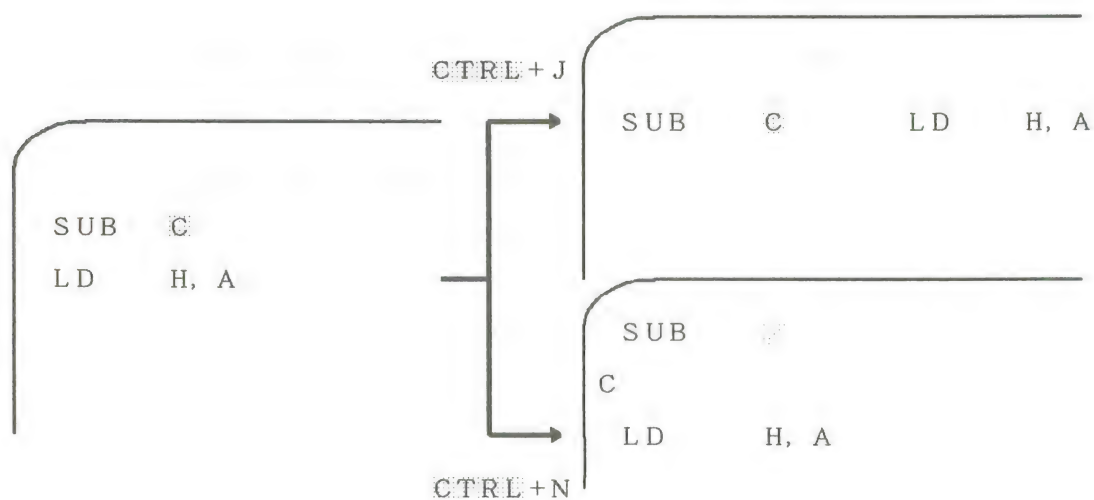


図 23

4.4.7 サーチ（検索）、リプレース（置換）に関するコマンド

《入力形態》 (ESC + [個数 | ? | *] +) F3 + <文字列>
+ { ↓ | ↑ | RETURN }

《機能》 文字列をサーチ（検索）します。

《説明》 ESC + 個数 ----- その個数番目のTargetをサーチします。

ESC + ? ----- Targetを1つずつサーチします。

ESC + * ----- テキスト中の最後のTargetをサーチします。

コマンド入力後、下に示すメッセージが出力されます。

target : <検索する文字列（20文字以内）>

（注）文字列には大文字、小文字の区別があります。

この文字列を指定する際には、マスクをかけることができます。

《例》

TargetにL??Pと指定した場合は、LOOP、LAMP等先頭がLで
最後がPの4文字の文字列がサーチされます。

この時、?自体を指定したい場合は、??と入力して下さい。

また、?自体を指定したい場合は、??と入力して下さい。

指定文字列を入力し、↓と入力すると前方へ（テキストの末尾方向へ）の、
↑と入力すると後方へ（テキストの先頭方向へ）のサーチが実行され、
カーソルは図24のようにサーチした文字列の先頭に移動します。
（RETURNの場合は、前回と同方向へのサーチになります。）

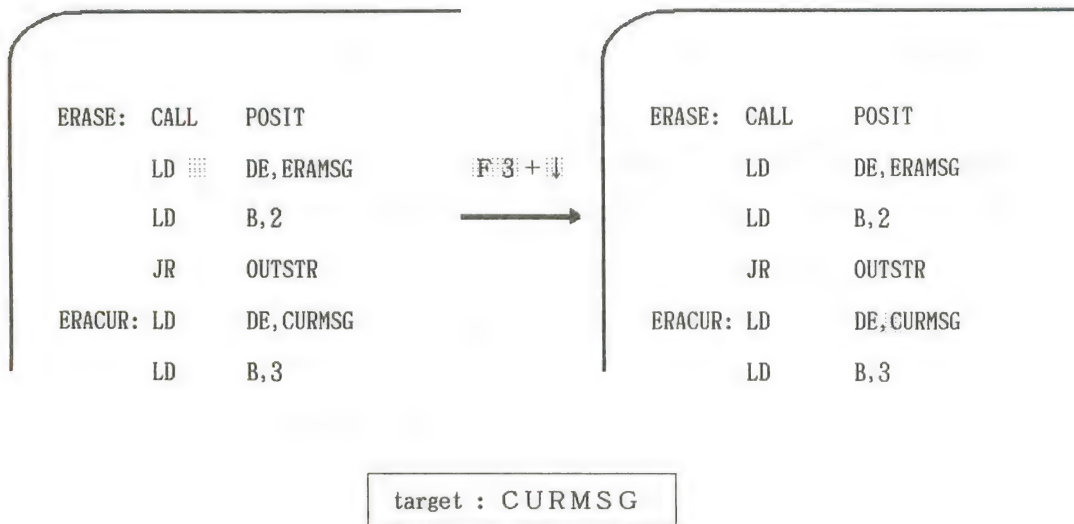


図 2 4

実行後の出力メッセージについては、次の通りです。

- ・テキストファイルの先頭及び終端に来て、もうこれ以上はないという場合、
" Not found " というメッセージが表示されます。
- ・サーチの回数を指定し、1つも見つけれなかった場合は" Can't find "と
というメッセージが表示されます。
- ・Targetが存在した場合は、次のようなメッセージが表示されます。
" △△△△ / ○○○○○ Searched "
(実際の個数) (指定個数)
ESC+ ? (あるいは*) の場合は、指定個数の部分は?と表示されます。

《注意》 次の場合、文字列を入力せずにRETURNを押すと () 内の文字列が
サーチされることになります。

- ① 以前サーチが実行された場合 (サーチ実行済の文字列)
- ② 以前リプレースが実行された場合 (リプレース実行済の置換元の文
字列)

《入力形態》 `[ESC + [個数 | * | ?] +] F4`

`+<文字列1>+<文字列2>+RETURN`

《機能》 文字列をリプレース（置換）します。

《説明》 コマンド入力後、下に示すメッセージが順次出力されます。

① Target: <置換される文字列（文字列1、20文字以内）>

② To: <置換する文字列（文字列2、20文字以内）>

（注）文字列には、大文字・小文字の区別があります。

指定文字列を入力し、RETURNを押してください。

この文字列を指定する際には、マスクをかけることができます。

《例》

Target を L??P, To を CALL とした場合、Target としては
LOOP, LAMP 等が当てはまり、次のように置換されます。

LOOP COOL

LAMP CAML

つまり、Target の ? の部分は置換されず、以前のまま残ります。

この時、? 自体を指定したい場合は、?? と入力して下さい。

?? 自体を指定したい場合は、??? と入力して下さい。

これは、文字列のマスクを除き、置換する文字列にも当てはまります。

また、置換する文字列を \$ とすると、何にも置き換えられず、Target の
消去になります。

F4 のみの場合は、1 つずつの置換を行います。

ESC + 個数 + F4 と入力した場合は、指定された個数分だけを置換し
ます。

個数の部分に * と入力すると、存在するすべての target を置換します。

? と入力した場合は、置換を行う前にその target を置換するかどうかの
メッセージがその都度出力されます。（図25参照）

△△△△/? Change OK ? (Y/N)

図 2 5

この時、Y, RETURNはYesとして、その他のキーはNoとして
みなされます。

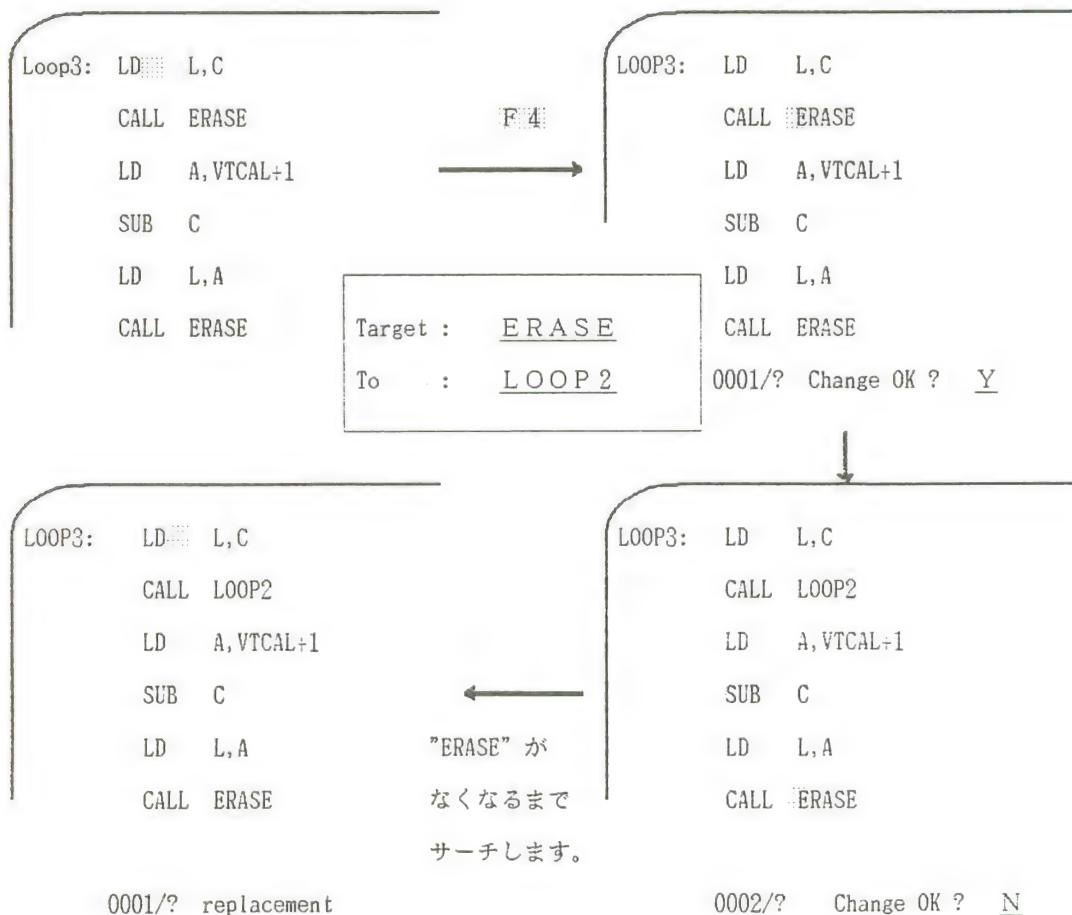


図 2 6

コマンド実行後の出力メッセージの内容は、下に示す通りです。

個数指定の場合。

・リブレース有り △△△△ / ○○○○○ replacement

(置換文字数) (指定文字数)

？あるいは*と入力した場合は、指定文字数の部分が？と表示されます。

・リブレース無し Non Replacement (Replace を1つも行わなかった場合)

Can't find (Targetが1つもなかった場合)

・テキストファイルの先頭及び終端に来て、もうこれ以上はないという場合、

" N o t f o u n d " というメッセージが表示されます。

《注意》 ・置換の結果、テキストバッファの容量が不足する場合、F4は実行されず、

" Over text buffer size " というメッセージを出力して終了します。

・Target, To の文字列を指定せず、RETURNのみを押した場合は、前回実行された文字列が使用されます。

《入力形態》 〔ESC+〔個数！？！*〕+〕 { F5 | CTRL+Q }

《機能》 サーチ・リブレースの再実行を行います。

《説明》 前回実行されたサーチ又はリブレースのコマンドを同一の条件で再実行します。

個数等を指定しなかった場合は、1回のみ再実行になります。

このコマンドの入力以前にサーチを行っていたらサーチ、リブレースを行っていたらリブレースを再実行します。

4.4.8 ブロック移動に関するコマンド

《入力形態》 (ESC+) { SELECT | CTRL+X } +

[{ COPY | DELETE | BUFFER | WRITE | APPEND }] + RETURN

《機能》 移動するブロックの範囲を指定します。

《説明》 SELECT (あるいはCTRL+X) キーを押すと、下に示すメッセージが順次出力されます。

① Please set start point

指定するブロックの始点 (終点) にカーソルを移動し、RETURN キーを押して下さい。その位置に@マークが表示されます。

行番号で指定したい場合は、始点となる行番号を入力してください。この時、*と入力すると、そのテキストの先頭行がセットされます。

② Please set end point

指定するブロックの終点 (始点) にカーソルを移動し、RETURN キーを押して下さい。その位置に新しい@マークが表示されます。

行番号で指定したい場合は、終点となる行番号を入力してください。この時、*と入力すると、そのテキストの最終行がセットされます。

①、②において、@マークにより範囲の指定を行いますが、この際には@マークの含まれる行全体が指定されます。

③

COPY DELETE BUFFER — SPACE —

または

WRITE APPEND — SPACE —

(この2つの表示は、SPACEキーによって切り換えられます。)

①, ②でマークした@によってはさまれるブロックをどのように処理するかを選択します。

- C) そのブロックを他の場所にコピーします。
- D) そのブロックを消去します。
- B) そのブロックをヤンクバッファに記憶します。
- W) そのブロックをフロッピーディスクに書き込みます。
- A) そのブロックをフロッピーディスク中のファイルに付加します。

RETURNのみを押した場合は、Bを押したものとみなされます。

このコマンド中であっても、インダイレクトコマンド (READ, NEW, REPLACE, SEARCHを除く) を使用することができます。

以下、それぞれのコマンド (C, D, B, W, A) について説明します。

《入力形態》 { SELECT | CTRL+X } + C + RETURN

《機能》 指定されたブロックを、ヤンクバッファへ記憶するとともに、カーソル位置へコピーします。

《説明》 コマンドを入力すると、次のメッセージが出力されます。

Please set copy point

カーソルをコピーしたいポイントに移動して、RETURNキーを押して下さい。

下図のようになります。

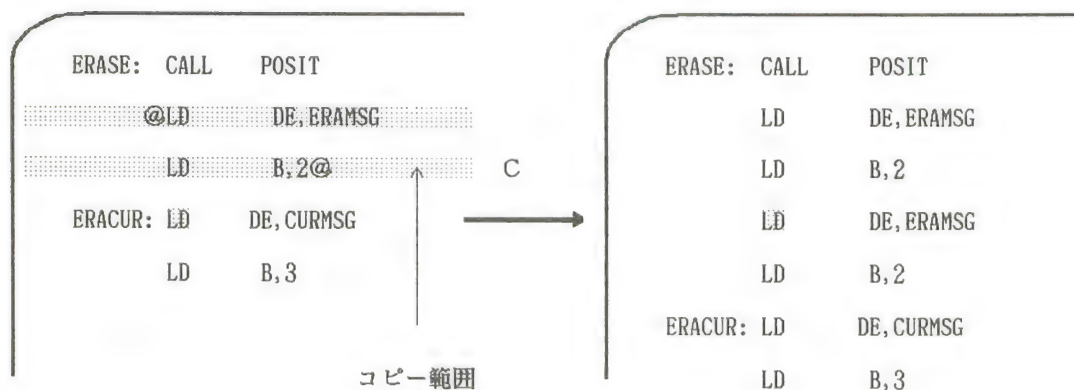


図 2 7

《注意》 コピーの結果、テキストバッファの容量を超えてしまうような場合は、コピーを実行せず、下記のエラーメッセージを出力し、終了します。

"Over text buffer size"

《入力形態》 { SELECT | CTRL+X } + D + RETURN

《機能》 指定されたブロックをヤンクバッファへ記憶するとともに、消去します。

《説明》 下図のようになります。

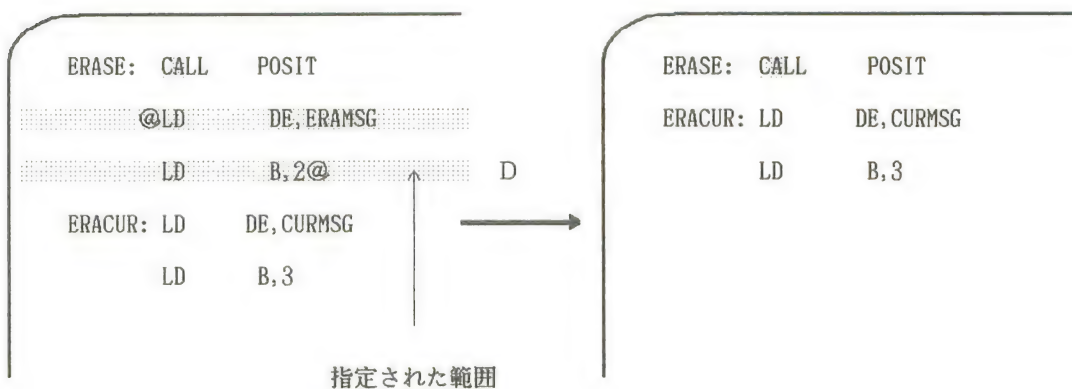


図 2 8

《入力形態》 { SELECT | CTRL+X } + B + RETURN

《機能》 指定された範囲をヤンクバッファに記憶します。

《説明》 これによって記憶されたブロックは、CTRL+Pによって任意の場所に貼り込みすることができます。

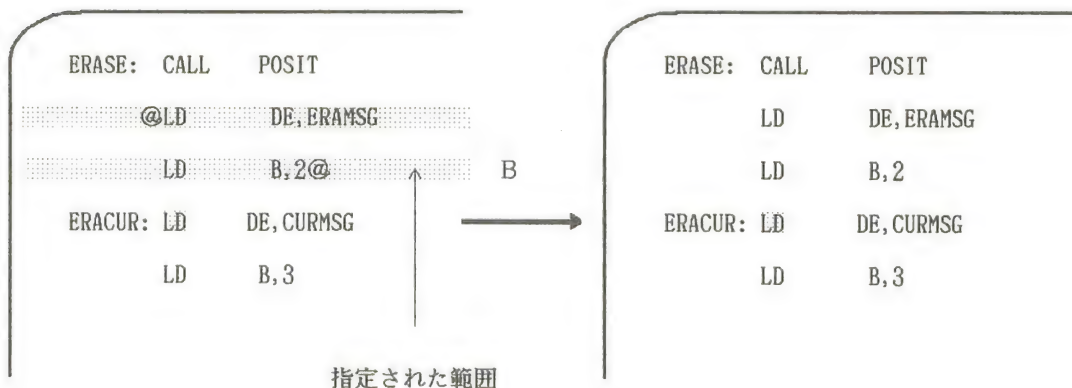


図 2 9

ヤンクバッファに、上図の指定されたブロックが記憶されました。

《入力形態》 { SELECT | CTRL+X } + W + RETURN

《機能》 エディットバッファ内のテキストファイルをフロッピーディスク上に書き込みます。

《説明》 コマンド入力後、下に示すメッセージが順次出力されます。

① Default name = デフォルト名出力

② File name : [<ファイル名>]

指定項目を入力して、RETURNを押してください。

《例》

Default name = SAMPLE. ASM
File name : _____ (RETURNのみ)

図 3 0

- 《注意》
- ① Default nameは、新規プログラム作成の場合には、tempとなります。
 - ② ファイル名を指定しないと、デフォルトファイルがとられます。
 - ③ 次の場合、エラーメッセージ出力後、スクリーンエディタに戻ります。
 - ・フロッピーディスク容量が不足している。
 - ・指定ドライブがセットされていない。

コマンド入力後、実際にWRITEを実行している間、" Now file writing" というメッセージが出力されます。

《入力形態》 { SELECT : CTRL+X } + A + RETURN

《機能》 エディットバッファ内のテキストファイルを、フロッピーディスク上のファイルにアペンド（付加）します。

《説明》 コマンド入力後、下に示すメッセージが順次出力されます。

① Default name = デフォルト名出力

② File name : [<ファイル名>]

指定項目を入力して、RETURNを押してください。

《例》

Default name = SAMPLE. ASM
File name : _____ (RETURNのみ)

図 3 1

- 《注意》
- ① 存在しないファイルを指定した場合は、WRITEと同じ機能になります。
 - ② ファイル名を指定しないと、デフォルトファイルがとられます。
 - ③ 次の場合、エラーメッセージ出力後、スクリーンエディタに戻ります。
 - ・フロッピーディスク容量が不足している。
 - ・指定ドライブがセットされていない。

APPENDを実行している間、“Now file writing”というメッセージが出力されます。

4.4.9 入出力に関するコマンド

《入力形態》 `ESC + { READ | R } + RETURN`

《機能》 フロッピーディスク上のテキストファイルをエディットバッファに読み込みます。

《説明》 コマンド入力後、下に示すメッセージが順次出力されます。

① `Default name = デフォルト名出力`

① `File name: <ファイル名>`

② `Start line: [<開始行>]`

③ `Number of lines: [<行数>]`

④ `Read lines(△△△△△-☆☆☆☆☆). (Y/N) ?`

(△△△△△=開始行, ☆☆☆☆☆=行数)

指定項目を入力して、`RETURN`を押してください。④のメッセージは、開始行と行数の確認メッセージです。Y（あるいは`RETURN`）を入力すると、`READ`が実行され、それ以外のキーを入力するとスクリーンエディタに戻ります。

《例》

```
Default name : SAMPLE. ASM
File name : _____ (RETURN のみ)
Start line: 0
Number of lines: _____ (RETURN のみ)
Read lines(00000-30000). (Y/N)? Y
```

図 3 2

《注意》

- ① ファイル名の中でドライブ名を省略すると、デフォルトドライブがとられます。また、指定したファイルがない場合は、エラーメッセージ出力後、スクリーンエディタに戻ります。
- ② 開始行を指定しないと、" 0 " が仮定されます。
- ③ 行数を指定しないか、あるいは、" 0 " を入力すると、最大長の" 3 0 0 0 0 " が仮定されます。
- ④ 指定した行数がファイルの行数を超えていると、ワーニングメッセージ出力後、スクリーンエディタに戻ります。

ファイルを読み込む直前のフリーエリアの大きさが、ファイルの大きさよりも小さい場合には、フリーエリアがなくなるまで読み込まれ、それ以降の部分は無視されます。

READを実行している間、"Now file reading"というメッセージが出力されます。

4.4.10 ステータス表示に関するコマンド

《入力形態》 `ESC+DIR+RETURN`

《機能》 ディレクトリを表示します。

《入力形態》 `ESC+DSKF+RETURN`

《機能》 フロッピーディスク諸元を表示します。

《例》

```
Drive is A
512      bytes / sector
2        sectors / cluster
354      clusters / diskette
185      clusters free
185      kbytes free
```

図33 ディスク諸元の一覧

《入力形態》 `ESC + { HELP | H } + RETURN`

`F6`

《機能》 ヘルプ・メッセージを表示します。

《説明》 コマンド入力後、下記の順番にメッセージが出力されます。

- ① ダイレクトコマンド一覧
- ② インダイレクトコマンド一覧
- ③ サーチコマンド一覧
- ④ ブロック移動コマンド一覧
- ⑤ ファンクション・キー一覧

《例》

【 Function key summary 】

```
F01 : Go to top of file
F02 : Go to tail of file
F03 : Search strings
F04 : Replace strings
F05 : Retry search & replace

F06 : Display help message
```

Hit any key !!

図34 ファンクション・キー一覧

ヘルプメッセージがまだ続く場合は、画面最下行に――more――
というメッセージが表示されます。

4.4.11 エディタの終了、初期化に関するコマンド

このコマンドを入力すると、誤操作によるテキスト破壊を防止するため、確認メッセージを出力します（図35，38参照）。メッセージに対してY（又はRETURN）以外のキーを入力すると、コマンドは実行されずにスクリーンエディタに戻ります。

《入力形態》 `ESC+NEW+RETURN`

《機能》 エディットバッファをクリアします。

《説明》 コマンドを入力した後、以下の確認メッセージが出力されます。

NEW? (Y/N)

図35

Y（又はRETURN）を入力すると、次のメッセージが表示されます。

File name: [<ファイル名>]

図36

ファイル名を入力すると、そのテキストファイルを新しく読み込みます。
ファイル名を入力せず、RETURNのみを押した場合は、NEWコマンドを終了します。

《入力形態》 `ESC + { QUIT | Q } + RETURN`

《機能》 エディタを終了します。

《説明》

- ① テキストの内容が変更されている場合

セーブ処理を行うかどうかのメッセージが出力されます。

SAVE ? (Y/N)

図 3 7

Y (又はRETURN) を入力するとセーブを行い、それ以外のキーを入力するとセーブを行いません。CTRL+CやESCのキーを入力するとQUIT処理を中止することが出来ます。セーブ後DOSのコマンドモードになります。

- ② テキストの内容が変更されていない場合

DOSのコマンドモードになります。

- ③ 一時バッファファイルはDOSへ復帰した場合のみ削除されます。

《入力形態》 `ESC + { UPDATE | U } + RETURN`

《機能》 ファイルの更新を行います。

《説明》

更新処理を行うかどうかのメッセージが出力されます。

UPDATE ? (Y/N)

図38

Y（又はRETURN）を入力すると更新を行い、それ以外のキーを入力すると更新を行いません。ファイルの更新が終了すると編集画面に戻ります。

5 付録

5.1 スクリーンエディタ (MED) のメッセージ

スクリーンエディタ実行時、MSX-DOSスクリーンエディタで発生したエラーメッセージの一覧表を表2に示します。

メッセージ	意 味
Can't access	ファイルがアクセスできません。
Disk I/O error	ディスクの入出力時にエラーが発生しました。
Disk full	ディスクがいっぱいです。
Disk write protected	ディスクが書込禁止の状態になっています。
Drive not ready	ディスクドライブがアクセスできません。
File not found	ファイルが見つかりません。
Invalid drive specification	ドライブの指定が正しくありません。
Invalid text	指定されたファイルが正しいテキストファイルではありません。
Not found	有効な文字列が1つも見つかりません。
Can't find	指定された文字列が1つも見つかりません。

メッセージ	意 味
Not so many lines	ファイルの行数が指定された開始行数よりも少ないため、ロードできません。
Over text buffer size	置換（あるいは貼り込み）を実行すると、テキストバッファの容量を超えてしまいます。
interrupt	文字列の検索・置換が中断されました。
Target not specified	文字列が指定されていません。
New file	起動時に指定されたファイルは新規作成です。

表2 スクリーンエディタのメッセージ

utility software

ユ ー テ ィ リ テ ィ ・ ソ フ ト ウ ェ ア

manual

マ ニ ュ ア ル

ユーティリティ・ソフトウェア・パッケージ

ユーティリティ・ソフトウェア・パッケージは 8080 系、および Z 80 系 CPU 向けのアセンブリ言語プログラム系の開発システムであり、大型コンピュータ上のソフトウェアに匹敵する強力な機能を持っています。また、MSX・L-80 リンクローダはマイクロソフト社のすべてのコンパイラ言語のプログラム作成時に使用します。ユーティリティ・ソフトウェア・パッケージは次の 4 つのプログラムから構成されています。

- MSX・M-80 マクロアセンブラ
- MSX・L-80 リンクローダ
- CREF-80 クロスリファレンサ
- LIB-80 ライブラリマネージャ

MSX・M-80 マクロアセンブラ

8080 系、Z 80 系の両方のニモニックコードをサポートし、リロケートブルなオブジェクトコードを作成するアセンブラ。

MSX・L-80 リンクローダ

アセンブル、あるいはコンパイル後のオブジェクトモジュール(.REL)を実行可能プログラム(.COM ファイル)に変換します。また、複数のモジュールを連結して 1 つの実行可能プログラムにすることができます。

CREF-80 クロスリファレンサ

MSX・M-80 によって作成されたクロスリファレンス・ファイルを入力し、ソースプログラム内の記号の参照、被参照を表すクロスリファレンス・リストを作成します。これによってデバッグを容易にすることができます。

LIB-80 ライブラリマネージャ

LIB-80 ライブラリマネージャはランタイムライブラリ・マネージャとして設計されましたが、また、LIB-80 はアセンブル、またはコンパイル済みのオブジェクトモジュールを集めてライブラリイメージにすることができます。

目 次

1. はじめに

1.1 本書の構成	2
1.2 システムの環境	3
1.3 提供するソフトウェア	3
1.4 ドキュメント	3
1.5 本書での表記法	4

2. MSX・M-80を使ったプログラム開発

2.1 用語の説明	6
2.2 MSX・M-80を使ったプログラム開発	7
2.3 MSX・M-80の特徴	10
2.3.1 2つのアセンブリ言語	10
2.3.2 リロケータビリティ	10
2.3.3 マクロ機能	10
2.3.4 条件付きアセンブル	10

3. MSX・M-80マクロアセンブリ言語

3.1 ソースプログラムの作成	12
3.1.1 ソースファイルの編成	12
3.1.2 ステートメント	12
3.2 シンボル	14
3.2.1 ラベル	15
3.2.2 パブリックシンボル	15

3.2.3	エクスターナルシンボル	16
3.2.4	シンボルとモード属性	17
3.2.5	プログラムカウンタ・モード	18
3.3	オペレーション	19
3.3.1	オペコード	19
3.3.2	擬似命令	19
3.3.3	マクロ	19
3.4	アーギュメント(式)	20
3.4.1	オペランド	20
3.4.2	演算子(オペレータ)	25
3.5	アセンブラの機能	29
3.6	擬似命令	29
■	命令セットの選択	29
	.Z80 30	.8080 30
■	データ定義およびシンボル定義	31
	DB/DEFB/DEFM 32	DC 33
	DS/DEFS 34	DW/DEFW 35
	EQU 36	
	EXT/EXTRN/EXTERNAL 37	
	BYTE EXT/BYTE EXTRN/BYTE EXTERNAL 37	
	ENTRY/GLOBAL/PUBLIC 38	
	SET/DEFL/ASET 39	
■	PCモード	40
	ASEG 41	CSEG 42
	DSEG 43	COMMON 44
	ORG 45	
	.PHASE/.DEPHASE 46	

■ ファイル関係 ————— 47

.COMMENT 48 END 49
INCLUDE/\$INCLUDE/MACLIB 50
NAME 51 .RADIX 52
.REQUEST 53

■ リスティング擬似命令 ————— 54

PAGE/\$EJECT 55 TITLE 55
TITLE 55
SUBTTL/\$TITLE 56
.LIST 57 .XLIST 58
.PRINTX 59 .SECOND 60
.LECOND 60 .TFCOND 61
.LALL 62
.XALL 62 .SALL 62
.CREF 63 .XCREF 63

3.7 マクロ機能 ————— 64

■ マクロの定義 ————— 64

MACRO~ENDM 65

■ マクロの呼び出し ————— 66

■ 反復擬似命令 ————— 68

REPT~ENDM 69 IRP~ENDM 70
IRPC~ENDM 72 EXITM 73
LOCAL 74

■ 特殊なマクロ演算子 ————— 75

& 76 ; ; 77
! 78 % 79

3.8 条件擬似命令	80
------------	----

IF×××××~ELSE~ENDIF 81

COND~ELSE~ENDC 81

4. MSX・M-80の実行

4.1 MSX・M-80	86
4.2 MSX・M-80で使用するファイル	87
4.3 MSX・M-80の呼び出し	88
4.4 MSX・M-80コマンド	89
4.4.1 MSX・M-80コマンドの形式	89
4.4.2 ファイル名の指定	89
4.4.3 MSX・M-80コマンドの例	91
4.5 スイッチ	93
4.6 メッセージ	94
4.7 リスティングフォーマット	95
4.8 エラーコードとエラーメッセージ	99

5. MSX・L-80リンクローダ

5.1 MSX・L-80	102
5.2 MSX・L-80で使用するファイル	103
5.3 MSX・L-80の呼び出し	104

5.4	MSX・L-80コマンド	105
5.4.1	MSX・L-80コマンドの形式	105
5.4.2	ファイル名の指定	105
5.4.3	ローディングの順序	105
5.4.4	MSX・L-80コマンドの例	106
5.5	スイッチ	107
5.6	メッセージ	117
5.7	エラーメッセージ	117

6. CREF-80クロスリファレンサ

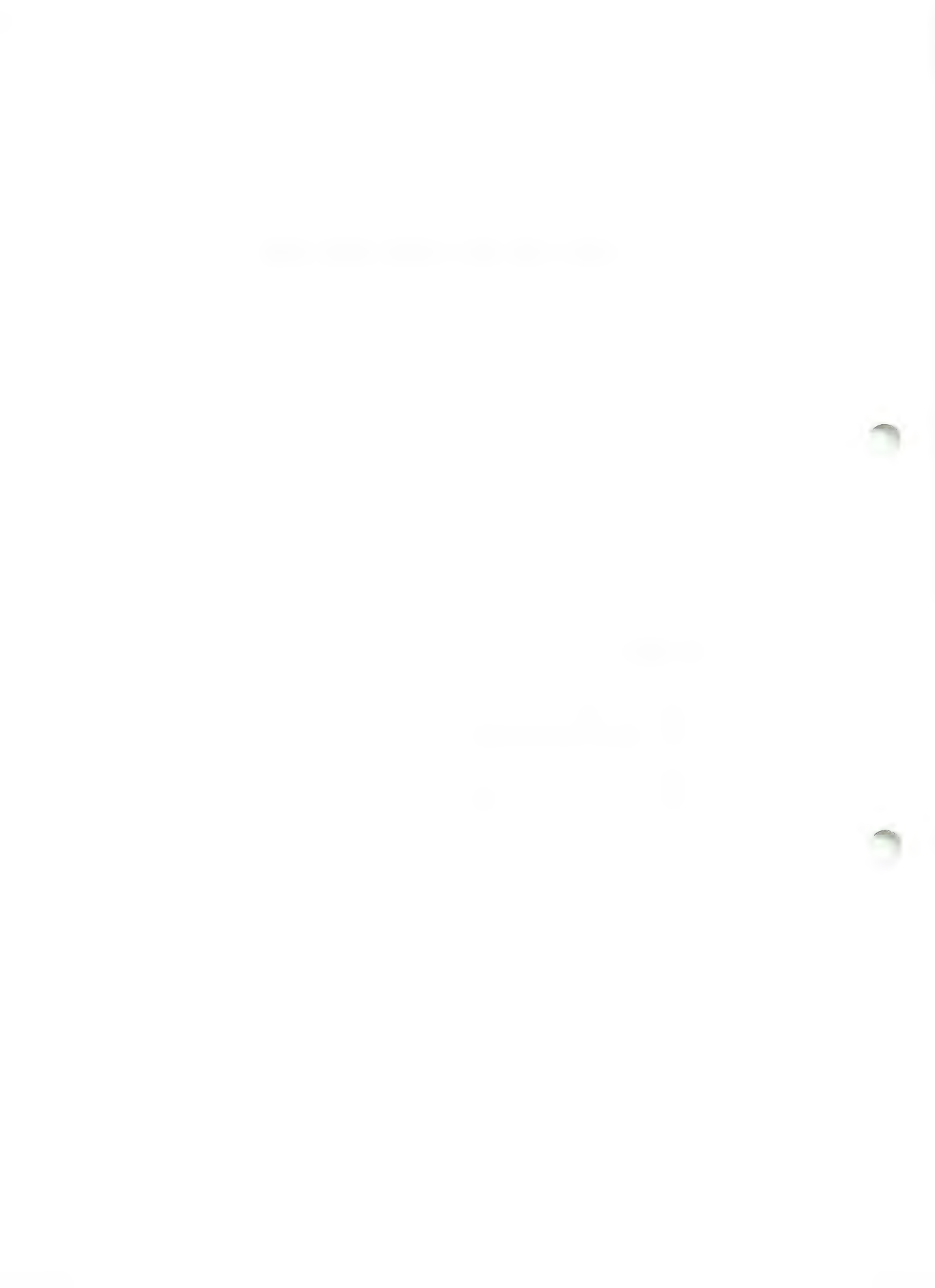
6.1	CREF-80	120
6.2	CREF-80で使用するファイル	120
6.3	クロスリファレンス・ファイルの作成	121
6.4	CREF-80の呼び出し	121
6.5	CREF-80コマンド	122
6.5.1	CREF-80コマンドの形式	122
6.5.2	ファイル名の指定	122
6.5.3	CREF-80コマンドの例	123
6.6	リスティング制御擬似命令	124
6.7	リスティングフォーマット	125

7. LIB-80ライブラリマネージャ

7.1	LIB-80	128
7.2	LIB-80で使用するファイル	129
7.3	LIB-80の呼び出し	130
7.4	LIB-80コマンド	130
7.4.1	LIB-80コマンドの形式	130
7.4.2	ファイル名の指定	135
7.4.3	LIB-80コマンドの例	135
7.5	スイッチ	136
7.6	注意事項	137

付録

付録 A	ASCIIキャラクタコード表	140
付録 B	MSX・M-80擬似命令表	141
付録 C	オペコード一覧表	145
付録 D	オブジェクトファイルの形式	150
索引		153



第 1 章

はじめに

- 1.1 本書の構成
- 1.2 システムの環境
- 1.3 提供するソフトウェア
- 1.4 ドキュメント
- 1.5 本書での表記法

第1章 はじめに

1.1 本書の構成

本書はユーティリティ・ソフトウェア・パッケージの使用法について詳しく解説しますが、アセンブリ言語のプログラミングとアセンブラに関する経験があるものとして解説します。アセンブラに関する知識を得た上で本書を読むことをおすすめします。

本書の構成は次のとおりです。

第1章 はじめに

ユーティリティ・ソフトウェア・パッケージの概略を解説します。

第2章 MSX・M-80を使ったプログラム開発

MSX・M-80を使ったプログラム開発について説明します。

第3章 MSX・M-80 マクロアセンブリ言語

MSX・M-80 マクロアセンブリ言語について説明します。

第4章 MSX・M-80の実行

MSX・M-80 マクロアセンブラの使用法（アセンブル方法）について説明します。

第5章 MSX・L-80 リンクローダ

MSX・L-80 リンクローダの働きと使用法を説明します。

第6章 CREF-80 クロスリファレンサ

CREF-80 クロスリファレンサの働きと使用法を説明します。

第7章 LIB-80 ライブラリマネージャ

LIB-80 ライブラリマネージャの働きと使用法を説明します。

付録

ASCII キャラクタコード、オブジェクトファイルのフォーマット、MSX・M-80 擬似命令、オペコード一覧表を記載します。

1.2 システムの環境

ユーティリティ・ソフトウェア・パッケージはそれぞれ次の大きさのメモリを必要とします。

パッケージ	メモリ
MSX・M-80	19 K バイト + 4 K バイト (バッファ)
MSX・L-80	14 K バイト
CREF-80	6 K バイト
LIB-80	5 K バイト

したがって、このユーティリティ・ソフトウェア・パッケージを使うためには 29 K バイト以上のメモリを持つシステムでなくてはなりません。また、1 台のディスクドライブでも使用可能ですが 2 台のディスクドライブを使用することをおすすめします。

1.3 提供するソフトウェア

本パッケージではユーティリティ・ソフトウェアとして以下の 4 つのファイルを提供しております。

M 80 . COM.....MSX・M-80 マクロアセンブラ

L 80 . COM.....MSX・L-80 リンクローダ

CREF 80 . COM.....CREF-80 クロスリファレンサ

LIB 80 . COM.....LIB-80 ライブラリマネージャ

1.4 ドキュメント

提供するドキュメントは本書、「ユーティリティ・ソフトウェア・マニュアル」です。

1.5 本書での表記法

本書ではコマンドおよびステートメントを表記する場合、以下の表記法を使用します。

- 大文字** 示されたとおりに入力しなければならないパラメータやコマンドを表します。
- < > (山形かっこ)** このかっこ内のテキストはある種類のパラメータを指定することを表します。ここに記述するパラメータは、そのパラメータの種類に応じた規則に従います。例えば <ファイル名> にはファイル名の規則に従った名称を入れなければなりません。
- [] (角形かっこ)** このかっこ内のパラメータがオプションであることを表示します。例えば <ファイル名> [, <ファイル名>] は、1つまたは2つのファイル名の入力を表します。
- ... (ピリオド3つ)** この3つのピリオドの前にある項目を必要な回数だけ入力することを表します。例えば <ファイル名> ... は、1つ以上のファイル名の入力を表します。
- 大文字が 内にある場合、CONTROL+C, RETURN のようなコントロール文字の入力を示します。
- { }** 大かっこは、複数個の項目からどれか1つを選択することを示します。大かっこで囲まれている項目は角形かっこで囲まれたものでない限り、少なくとも1つは入力します。

第2章

MSX・M-80を使ったプログラム開発

2.1 用語の説明

2.2 MSX・M-80を使ったプログラム開発

2.3 MSX・M-80の特徴

第2章

MSX・M-80を使ったプログラム開発

2.1 用語の説明

MSX・M-80 を使ったプログラム開発の説明の前に、プログラム開発についての解説で一般に使用されている用語を説明します。

ソースプログラム	汎用テキストエディタで作成したプログラムテキストステートメントで構成されています。
ソースファイル	ソースプログラムが入っているファイル。ソースファイルはアセンブラ、またはコンパイラの入力ファイルであり、ASCII フォーマットでなければなりません。マクロアセンブラのソースファイルの拡張子の省略値は .MAC です。
オブジェクトプログラム (オブジェクトモジュール)	ソースプログラムをアセンブラまたはコンパイラで翻訳した結果。リロケート可能な機械語に変換されています。
オブジェクトファイル	オブジェクトプログラムが入っているファイル。オブジェクトファイルはコンパイラの出力ファイルであり、また、リンクローダの入力ファイルでもあります。オブジェクトファイルの標準的な拡張子は .REL です。
モジュール	アセンブラ、コンパイラまたはリンクローダが扱うことのできる基本的な命令語コード群の単位です。モジュールには開発の段階や性質に応じて幾つかの種類があります。アセンブラやコンパイラの翻訳結果であるオブジェクトプログラムはオブジェクトモジュールと呼ばれますが、その性質からはリロケート可能なモジュールとも呼びます。このモジュールがリンクローダによって処理された最終的な実行可能プログラムは実行可能モジュールと呼ばれます。
外部参照 (グローバルリファレンス)	異なるモジュール間でシンボルを参照することを外部参照（グローバルリファレンス）と言います。リンクローダは参照関係のあるモジュールを連結し、グローバルリファレンスの解決をします。また、グローバルリファレンスは他のモジュールから参照されるシンボルのことを指すこともあります。
エントリポイント	他のモジュールが呼び出すことのできるラベル。パブリックシンボルはエントリポイントとして使うことができます。

リロケータブル

あるモジュールを、メモリ内の別の場所にリロケート（再配置）して実行することができる場合、そのモジュールはリロケータブル（再配置可能）であると言えます。リロケータブルモジュールでは、ラベルまたは変数がそのモジュールの先頭アドレスからの相対アドレスで表わされています。これらのラベルや変数を“コードリラティブ”であると言い、モジュールがリンクローダによってロードされる時に絶対アドレスに変換されます。

リンクロード

リンクローダ（MSX・L-80）がリロケータブルモジュール内のラベルや変数の絶対アドレスを計算したり、他のモジュールやランタイムライブラリを検索してグローバルリファレンスを解決したりする間の処理を言います。ロードとリンクを行ったあと、リンクローダはメモリ内にロードしたモジュールを単一の COM ファイルとして、ディスクに保存します。

2.2 MSX・M-80を使ったプログラム開発

アセンブリ言語プログラムの開発の概要を図 2-1 に示します。また、図 2-1 の説明を以下に説明します。

- (1) MSX-C パッケージ付属のスクリーンエディタ、または他の汎用テキストエディタを使用してアセンブリ言語のソースプログラムを作成します。
- (2) MSX・M-80 アセンブラでソースプログラムをアセンブルし、中間オブジェクトコードを含むオブジェクトプログラムをオブジェクトファイルに作成します。
中間オブジェクトコードはソースコードに比べて機械語に近いものですが、まだ実行することはできません。
- (3) MSX・L-80 リンクローダで、個々にアセンブル済みのオブジェクトプログラム（複数の場合もある）をロード、リンクし、実行可能プログラムを実行可能ファイルに作成します。
リンクローダは、中間オブジェクト・コードのプログラムを実行可能な機械語から成るプログラムに変換します。プログラムの実行はオペレーティングシステムのコマンドと同じようなキーインで行うことができます。実行可能プログラムの入っているファイルをコマンドファイルと呼びます。
- (4) デバッグ時など必要に応じて CREF-80 クロスリファレンサを使い、クロスリファレンス・リスティングファイルを出力します。クロスリファレンス・リスティングファイルには MSX・M-80 のリスティング情報に加え、ラベルとその定義位置と参照している行のクロスリファレンス情報が出力され、デバックに役立ちます。
- (5) サブルーチンがたくさんあるプログラムなどの場合には、サブルーチンのオブジェクトモジュールを LIB-80 ライブラリマネージャでライブラリファイルに集め、MSX・L-80 でのコマンド入力を簡易にすることもできます。
- (6) MSX・L-80 で作成した実行可能プログラムは MSX-S BUG（別売）を使ってデバッグすることができます。

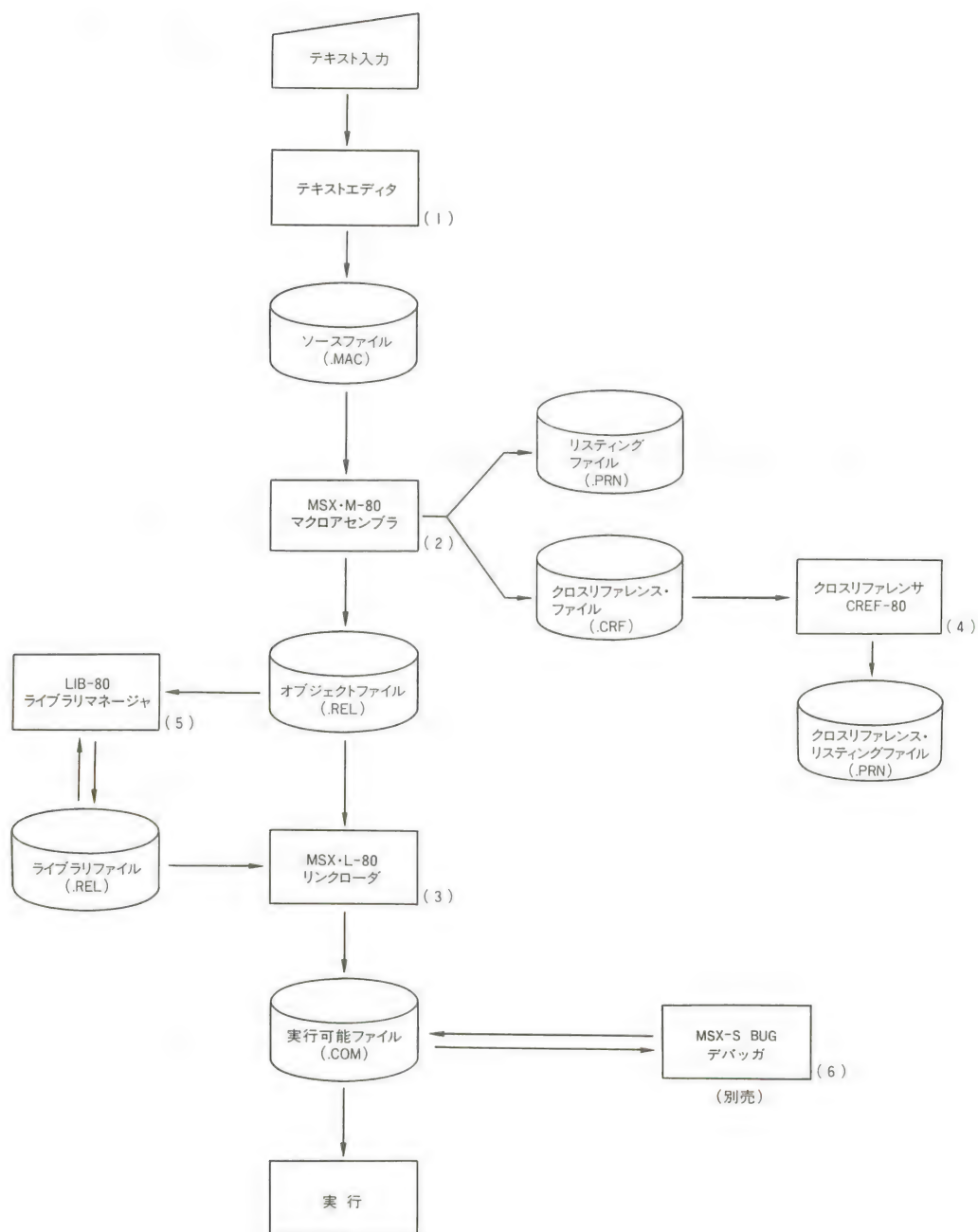


図 2-1 アセンブリ言語のプログラム開発

ユーティリティ・ソフトウェア・パッケージを使ったプログラム開発では次の応用をすることができます。

- プログラムのモジュール化

大規模なプログラムをいくつかに分割して作成することができます。

分割したそれぞれのプログラムをモジュールと呼びます。各モジュールのソースプログラムをそれぞれ別に作成、アセンブル、およびデバッグすることができるので、デバッグ時には正しく動作しないプログラムのみを修正し再アセンブルします。

プログラムをモジュール化すると大きなプログラムを複数人で開発することができ、また、正しく動作しない部分だけをくり返しアセンブルすることができるので開発時間を短縮することができます。

- モジュールのロード位置の任意な配置

第5章で説明する制限を除き、リンクローダを使ってモジュールをメモリ上の任意の位置へ配置することができます。

- 使用頻度の高いモジュールの共用

汎用性のある使用頻度の高い部分を独立したモジュールとしてアセンブルしておき、複数のプログラムで共用(MSX・L-80リンクローダを使って連結します)することにより、コーディングするプログラム量を小さくし、プログラミングの効率化を計ることができます。

- モードの指定による柔軟なメモリの使用

マクロアセンブラでは一つのモジュールを4つのモードに使い分けてアセンブルすることができます。絶対(Absolute)モード、データ相対(Data-relative)モード、コード相対(Code-relative)モード、COMMON相対(Common-relative)モードの4つです。

絶対モードではメモリ内の固定アドレスに配置するようにコードを生成します。

3つの相対モードの各々は、個々のセグメントにアセンブルされます。各セグメント内を参照するアドレスはセグメントの先頭バイトを0とした相対アドレスで表わし、MSX・L-80でロード時に指定したスタートアドレスからの固定アドレスに変換します。相対モードのコードはメモリ内でのリロケート(再配置)可能であり、この能力をリロケートビリティと呼びます。オブジェクトファイルの拡張子.RELはリロケートビリティに由来しています。

2.3 MSX・M-80の特徴

2.3.1 2つのアセンブリ言語

MSX・M-80は8080/Z80の両方のニモニックをサポートします。

2.3.2 リロケートビリティ

MSX・M-80はリロケートブルなコードのモジュールを作成することができます。また、多くのアセンブラと同様に絶対コードを作成することもできます。リロケートビリティの利点は、実行時の配置位置を意識せずにプログラムをモジュールごとにアセンブルすることができるということです。さらに、第5章に記述されているいくつかの制約を除いてそのモジュールをメモリ内の任意な位置に置くことができます。

リロケートブルモジュールはコーディングが容易で、テスト、デバッグ、および修正が速いという利点を持っています。また、あとでRAMまたはROM/PROMにロードするアセンブル後のオブジェクトコードのセグメントを指定することも可能です。リロケートビリティについては3.2.5のモードの項で詳しく述べます。

2.3.3 マクロ機能

MSX・M-80はインテル社の標準マクロ機能をサポートします。マクロ機能によって、頻繁に使用する命令セットのブロックに名前を定義し、命令セットの代わりに名前を書くことができます。この名前を書くことによって、同じ命令セットを何回もくり返しコーディングする必要がなくなります。

この命令セットのブロックをマクロと呼び、名前を付けることをマクロ定義と呼びます。マクロ定義はマクロを使用する前にソースプログラム上で行いますが、ディスク上の別のソースファイルにマクロ定義をしておき、INCLUDE 擬似命令で挿入するようにすると、マクロ定義を複数のモジュールで共有できるので便利です。プログラム中で命令セットが必要な場合はマクロ名だけを指定して、マクロを呼び出します。MSX・M-80はマクロ名の指定があると、マクロ定義しておいた命令セットをソースプログラム上に展開した形でアセンブルを行います。マクロの使用によりソースプログラムのサイズを減少することができます。

また、マクロの呼び出し時に、マクロの展開に使用するパラメータをアセンブラに対して指定することもできます。

マクロはネストする（重ねる）ことができます。すなわち、マクロは他のマクロ内から呼び出すこともできます。マクロのネスティングはメモリのサイズによってのみ制限を受けます。

2.3.4 条件付きアセンブル

MSX・M-80は条件付きアセンブルをサポートします。プログラマはプログラムのどの部分をアセンブルするか、あるいはアセンブルしないかの条件を指定することができます。条件付きアセンブル機能はアセンブルパスのテスト、シンボル定義、およびマクロへのパラメータなどの条件付き擬似命令によって拡張されています。条件は255レベルまでネストが可能です。

第3章

MSX・M-80マクロアセンブリ言語

- 3.1 ソースプログラムの作成
- 3.2 シンボル
- 3.3 オペレーション
- 3.4 アーギュメント
- 3.5 アセンブラの機能
- 3.6 擬似命令
- 3.7 マクロ機能
- 3.8 条件アセンブル機能

第3章

MSX・M-80マクロアセンブリ言語

3.1 ソースプログラムの作成

この章では MSX・M-80 マクロアセンブラのソースプログラムを作成するのに必要なことを解説します。ソースプログラムはテキストエディタを使用して作成します。

ソースファイルは、第4章に記述しているような手続きを経てアセンブルします。ソースプログラムのアセンブル方法は第4章で説明します。

3.1.1 ソース・ファイルの編成

MSX・M-80 マクロアセンブラのソースファイルはアセンブリ言語で書かれたステートメントを含む一連の行です。

- ファイルの最後の行は、END ステートメントにしなければなりません。
- マatchingステートメント（たとえば、IF...ENDIF）は、正しい順序で入力しなければなりません。
- ソースファイルから入力される各行は最大 132 文字の長さまで受け入れる事ができます。
- MSX・M-80 では小文字でタイプされたすべての記号、命令コード、擬似命令コードは大文字に変換されます。ただし引用符で囲まれた文字列や注釈は小文字から大文字への変換はされませんが、ビット 7 (MSB) が 0 にされるため、ひらがなカナ文字グラフィック文字の使用はできません。
- ソースファイルがエディタで与えられた行番号を含んでいる場合、その行番号の各バイトの上位ビットには 1 が設定されていなければなりません。

3.1.2 ステートメント

ステートメントの形式は次のとおりです。

シンボル フィールド	オペレーション フィールド	アーギュメント フィールド	コメントフィールド
---------------	------------------	------------------	-----------

例) <u>BUF :</u>	<u>DS</u>	<u>1000 H</u>	<u>; create a buffer</u>
↓ シンボル フィールド	↓ オペレーション フィールド	↓ アーギュメント フィールド	↓ コメント フィールド

シンボルフィールド

このフィールドにはシンボルを記述します。シンボルの直後には EQU, SET, DEFL, ASET または MACRO ステートメントを除いてコロン (:) を付けます。

オペレーションフィールド

このフィールドにはオペコード、擬似命令コード、マクロ名を記述します。

アーギュメントフィールド

このフィールドには式（定数、変数、レジスタ名、オペランドおよびオペレータ（演算子））を記述します。

コメントフィールド

このフィールドの先頭にはセミコロン (;) を置き、以降にコメントテキストを記入します。

すべてのフィールドはオプション（ユーザの選択にまかされていること）です。完全な空白行（何もかかないこと）にすることもできます。

ステートメントは任意のカラムから開始することができます。複数の空白またはタブをフィールド間に挿入して読みやすくすることができますが、各フィールド間には少なくとも1つの空白またはタブが必要です。

注 意 ステートメントの基本的な構成はオペレーションおよびそのアーギュメント（引数）です。

コメントはセミコロンで始まりキャリッジリターンで終わります。

コメントが長い場合は、COMMENT 擬似命令を使用して各行ごとのセミコロンの入力を省略することができます。COMMENT の解説については 47 ページを参照してください。

3.2 シンボル

シンボルは特定の値に対して定義する名前です。シンボルの定義方法は次の2通りあります。

- 機械命令やデータ定義をするステートメントにラベルとして定義します。シンボルの持つ値は機械命令やデータ領域のアドレスです。また、シンボルはシンボルを定義しているセグメントに対応するモード属性を持ちます。
- EQU や SET 擬似命令を使って定義します。シンボルの持つ値は EQU や SET 擬似命令で指定した〈式〉の値です。また、シンボルは〈式〉が持つ属性を持ちます。

定義したシンボルはモジュール内でアークギュメントフィールドの一項目として使用することができますが、パブリックシンボルとして宣言して他のモジュールから使えるようにしたり、また、エクスターナルシンボルとして宣言して他のモジュールで定義されたシンボルを使うこともできます。シンボルは次の規則に従います。

- シンボルの長さは任意ですが、リンクローダに渡される有効文字数はシンボルによって次のように異なります。
 - a. ラベルでは先頭の16文字までが有効です。
 - b. パブリックおよびエクスターナルシンボルでは、先頭の6文字までがリンクローダに渡されます。余分な文字は切り捨てられます。
- シンボルに使える文字は次の通りです。

A~Z 0~9 \$. ? @ -

- シンボルは数字で開始してはいけません。
- 小文字は大文字とみなしますので、シンボルは、名前を大文字または小文字のいずれで入力してもかまいません。

3.2.1 ラベル

ラベルは機械命令やデータ定義をしているステートメントのシンボルフィールドに書きます。

ラベルはラベルを定義しているステートメントの命令やデータ領域のアドレスを値として持っています。ラベルはプログラム中のステートメントでアーギュメントフィールドの一項目として使うことができる参照ポイントとなります。

ラベルはステートメントの最初のフィールドでなければなりません。また、ラベルはシンボルの規則に従う16文字までの名前です。ラベルの直後にはコロンを1つ付けます（コロンを2つ付けるとパブリックシンボルとなります。3.2.2を参照してください）。

〈ラベル名〉

例) BUF: DS 1000H

BUF: は1000 H バイトの予約スペースの先頭アドレスを値として持ちます。

定義されたラベルはアーギュメントフィールドの項目として使用することができます。アーギュメントにラベルを書くと、ラベルを定義しているステートメントを参照するアドレスに置き換わります。

STA BUF

前記の例ではアキュムレータの値がラベル BUF で表されるエリアにセットされます。

3.2.2 パブリックシンボル

パブリックシンボルはシンボルと同じように、アーギュメントに書くことができます。シンボルとの違いはパブリックシンボルが他のプログラムからも使えることです。パブリックシンボルの宣言方法は次の2通りあります。

- シンボルの直後にコロンを2つ付ける。

例) FOO:: RET

- 擬似命令の PUBLIC, ENTRY, または GLOBAL で宣言する。

例) PUBLIC FOO

これらの擬似命令については3.6のデータ定義とシンボル定義の擬似命令の項を参照ください。

上記2つの宣言方法の効果は同じです。次の2つの例の結果は同じになります。

例) FOO:: RET

例) PUBLIC FOO

FOO: RET

3.2.3 エクスターナルシンボル

エクスターナルシンボルは他のモジュールでパブリックシンボルとして定義されているシンボルです。エクスターナルシンボルはリンクロード時に対応する他のモジュールのパブリックシンボルの値に置き換わります。

例) モジュール 1

```
EXT          TEC      ;EXTERNAL TEC
:
CALL         TEC
```

モジュール 2

```
TEC: : LD          C, 2 ;PUBLIC TEC
```

モジュール 1 の TEC にはリンクロード時にモジュール 2 の LD C, 2 のアドレスが値としてセットされます。

例) モジュール 1

```
ENTRY       HWR      ;PUBLIC HWR
:
HWR EQU      7
:
```

モジュール 2

```
BYTE EXT    HWR      ;EXTERNAL HWR
:
ITE: DB      HWR      ;VALUE=7
```

モジュール 2 の HWR にはリンクロード時に 7 が値としてセットされるので、DB HWR で定義した 1 バイトの領域には 7 がセットされます。エクスターナルシンボルの宣言方法は次の 3 通りです。

- シンボルの直後にポンド符号 (##) を入れます。

例) CALL FOO##

FOO は他のプログラムで定義され、2 バイトの値を持つシンボルであることを宣言します。

- 擬似命令 EXT, EXTRN, または EXTERNAL のいずれかで他のプログラムのシンボルを参照することを宣言します。

例) EXT FOO

FOO は他のプログラムで定義され、2 バイトの値を持つシンボルであることを宣言します。

- 1 バイトの値に対しては擬似命令 `BYTE EXT`, `BYTE EXTRN`, または `BYTE EXTERNAL` のいずれかを使います。

例) `BYTE EXT FOO`

`FOO` は他のプログラムで定義された 1 バイトの値として宣言されます。

これらの宣言の効果は同じです。したがって次の 2 つの例の結果は同じになります。

例) `CALL FOO##`

例) `EXT FOO`

`CALL FOO`

3.2.4 シンボルとモード属性

アーギュメントフィールドにシンボルを記入すると、シンボルを定義したステートメントを参照します。アーギュメントフィールドのシンボルはシンボルの持つ値に置き換わり、オペレーションとして使用されます。

シンボルはカウンタ (PC) モードに従って評価されます。PC モードが絶対モードならば絶対アドレスで、コード相対モード、データ相対モードならば相対アドレスで、また、COMMON モードならば他のモジュールと共用するアドレスがそれぞれ設定されます。

3.2.5 プログラムカウンタ・モード

MSX・M-80では1つのモジュールを必要に応じて次の4つのプログラムカウンタ・モード(PCモード)に分けてアセンブルすることができます。擬似命令の詳細は3.6.3のPCモードの擬似命令を参照ください。

- 絶対モード (Absolute)

擬似命令: ASEG

絶対モードは実行時のメモリ上の配置位置を特定のアドレスに固定するプログラム部分について指定します。絶対モードのシンボルおよび式は絶対値を持ち、非リロケートブルです。

絶対モードのセグメント中のコードは非リロケートブルコードです。

- データ相対モード (Data-relative)

擬似命令: DSEG

データ相対モードは実行時にメモリ領域を書き換える可能性があるプログラム部分について指定します。したがって、データ相対モードのセグメントはRAM上にロードします。データ相対モードはプログラムのデータ領域に適しています。データ相対モードのシンボルはリロケートブルです。

- コード相対モード (Code-relative)

擬似命令: CSEG

コード相対モードは実行時にメモリ領域の書き換えの可能性がないプログラム部分について指定します。したがってROM/PROM上に置くこともできます。コード相対モードはプログラムのコード領域に適しています。データ相対モードのシンボルはリロケートブルです。

- COMMON 相対モード (Common-relative)

擬似命令: COMMON

COMMON モードは他のプログラムとの共通データ領域の部分について指定します。

COMMON モードのセグメントは他のプログラムと共用することができます。

モードの指定がない場合はコード相対モードを仮定します。相対モードのプログラム部分はリロケートブルであり、メモリの配置について柔軟なプログラミングをすることができます。

アセンブル時には4つの各モードに対応したセグメントごとにオブジェクトコードを生成します。また、リンクロードでは、指定がないかぎりデフォルトのアドレス 103 H から COMMON 相対セグメント、データ相対セグメント、コード相対セグメントを順に配置し、相対セグメント内に含まれている相対アドレスを絶対アドレスに変換します。アドレス 100 H~102 H にはプログラム開始アドレスへのジャンプ命令がセットされます。

複数のモジュールをリンクする場合には各モジュールごとに同じモードのセグメント同志を統合します。また、コード相対セグメントとデータ相対セグメントはMSX・L-80のスイッチ/Pおよび/Dを使ってセグメントを任意のアドレスに配置することもできます。

3.3 オペレーション

オペレーションフィールドには次の2つのオペレーションのどちらかを記述します。

- マクロ名
- オペコード/擬似命令

オペレーションフィールドの項目は上記の順序で評価されますので、オペコードや擬似命令と同じ名前のマクロ定義を作成するとマクロ定義の方が優先され、そのマクロ定義以降では同じ名前のオペコードや擬似命令を使うことができません。たとえば、マクロ名がADDというマクロ定義をすると、オペコードのADDを使用することができません。

ステートメント中にオペレーションがなくアークギュメントに式がある場合、MSX・M-80はエラーを表示せずバイト定義擬似命令を仮定します。

3.3.1 オペコード

オペコードは機械命令に対応するニモニック名です。MSX・M-80は8080/Z80の2つの機械語命令セットをサポートします。付録Cにオペコードの一覧表を簡単な機能の解説とともに記載します。8080/Z80のうちのどちらの命令セットでプログラムを作成するかは、擬似命令で選択することができますが、特に指定がなければ、8080を選択したものとみなされます。また、プログラムの途中から、異なる命令語セットにすることができます。詳細については3.6の命令セットの選択を参照ください。

3.3.2 擬似命令

擬似命令はマイクロプロセッサではなくアセンブラに対する指示です。MSX・M-80はアセンブラを指示する種々の機能を持つたくさんの擬似命令をサポートします。擬似命令の詳細は3.6に記しており、また、付録Bに要約したものを記載しています。

3.3.3 マクロ

マクロ名はマクロ定義ブロックに付けた名前です。頻繁に使用する命令セットのブロックをマクロ定義しておき、マクロ名を指定するだけで呼び出すことができます。

3.4 アーギュメント(式)

オペコードと擬似命令のアーギュメントには式を記述します。式は、たとえば $5+4*3$ のような数式に似ています。式はオペランド（たとえば数式内の 5, 4, 3）および演算子（オペレータ、たとえば数式内の +, *）から構成されます。式は 1 つ以上のオペランドを含み、2 つ以上のオペランドを含む場合はオペランドは演算子によって互いに関係づけます。

例) ADDR (7 * 2) 6-3 8/7 ADDR + 3 9 > 8

以下にオペランドとオペレータの形式を説明します。

3.4.1 オペランド

数値

数値のデフォルトの基数は、10 進数です。基数は、.RADIX 擬似命令によって 2 進から 16 進までの任意のベースに変更できます。基数が 10 より大きい場合には 9 の次の数には A~F を使用します。数値の最初の文字が数字でない場合には、その数の前には 0 がなければなりません。

数値は次の表記法が使用されていない限り、その時点で設定されている基数を使用します。

例) nnnnB	2 進
nnnnD	10 進
nnnnO	8 進
nnnnH	16 進
X'nnnn'	16 進

数値は 16 ビットの符号なしの 2 進数です。65535 (16 ビットで表すことができる最大値) を越える数のオーバーフローは無視され、その結果は 16 ビットで表わされる範囲内の数になります。

ASCII スtring

String (文字列) はゼロまたは 1 個以上の文字より構成され、引用符で囲みます。引用符は、シングル (') またはダブル (") のいずれかを使用します。引用符付 String がアークギュメントとして入力された時、引用符内の文字が順にメモリにストアされます。

例) DB "ABC"

4	1	4	2	4	3
0	1	2	3		

上の例では先頭のアドレスに A の ASCII コード (41 H)、2 番目のアドレスに B (42 H)、3 番目のアドレスに C (43 H) がストアされます。

また、シングルクォーテーション (') で囲んでいる String 中には、ダブルクォーテーション (") を String の一部として入れておくことができ、ダブルクォーテーション (") で囲んでいる String 中には、シングルクォーテーション (') を String の一部として入れておくことができます。

例) "I am 'great' today"
 'I am "great" today'

区切り記号の引用符を文字として用いたい場合、それが必要な部分をさらに引用符で囲みます。

例) "I am \"great\" today"

上の例は次のようにストアされます。

例) I am "great" today

引用符に文字がない場合、String はヌル String として評価されます。

文字定数

ストリングに似て、文字定数は0~2文字のASCII文字より構成され、引用符で囲みます。引用符はシングルでもダブルでも使用できます。引用符を文字として用いる場合、必要となる部分をさらに引用符で囲みます。ストリングとの違いは次のとおりです。

- 文字定数は文字数が0~2個です。
- 式が2つ以上のオペランドをもっている場合、引用符で囲まれた文字は文字定数として扱われます。引用符で囲まれた文字が式中唯一のオペランドである場合、その文字は、ストリングとして扱われます。

例) 'A' + 1.....文字定数

'A'ストリング

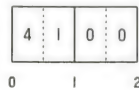
- 文字定数の値は計算され、結果は最初のアドレスに下位バイトが、2番目のアドレスに上位バイトがセットされます。

例) DW 'AB' + 0

上の例では文字定数が4142 Hと評価され、最初のアドレスに42を、2番目のアドレスに41をセットします。

1文字の文字定数はその文字のASCIIコードに対応する値をもっています。文字定数の上位バイトはゼロであり、下位バイトは文字のASCIIコードに対応する値です。たとえば定数'A'の値は41 Hです。

例) DW 'A'



上の例では、メモリ上に図のような2バイトの値が確保されます。なお、アセンブリリストのオブジェクトコード欄では、上位バイト、下位バイトの順に表示されます。

2つの文字の文字定数は、上位バイトに最初の文字のASCIIコードをもっています。たとえば文字定数'AB' + 0の値は41 H * 256 + 42 H + 0です。

例) DW 'AB' + 0



ASCII文字の10進数および16進数の値を付録Aに記載しています。

式内のシンボル

シンボルは式内のオペランドとして使用することができます。

シンボルは評価され、シンボルの持つ値がシンボルに代って置き換えられます。

- オペレーションはシンボルの値を使用して行われます。

オペランドとしてシンボルを使用する利点は、ジャンプ先やデータ領域のアドレス値を覚えておく必要がなく、シンボルの名前を覚えていればよいということです。シンボルの名前付けは覚えやすく意味を持った綴りが便利です。プログラム内の参照ポイントの数が多い場合、オペランドとしてシンボル名を使える事は大きな魅力です。

- 式中にエクスターナルシンボルを使用するための規則

- a. エクスターナルシンボルは次の演算子を式中に使用することができます。

+ - * / MOD HIGH LOW

- b. エクスターナルシンボルが式中で使用されていた場合は、その式の結果は外部参照となります。

- 式中のシンボルに関するモードの規則

- a. オペレーションが AND, OR, または XOR を除いて、オペランドは任意のモードでもかまいません。
- b. AND, OR, XOR, および SHR の場合、両オペランドは絶対モードかつ内部参照でなくてはなりません。
- c. 式が絶対モードのオペランドと他のモードのオペランドを含む場合、その式は絶対モード以外の他方のモードとなります。
- d. 異なるモードで2つのオペランドを除算した場合式は絶対モードとなり、同じモードの場合式はそのオペランドのモードとなります。
- e. データ相対シンボルおよびコード相対シンボルを加えた場合、式の値は未定です。MSX・M-80 はその式を未解決のものとして MSX・L-80 に渡し、MSX・L-80 が解決します。

カレント・プログラムカウンタ・シンボル

アークギュメントフィールドにはカレント・プログラムカウンタ・シンボルというものを書くことができます。カレント・プログラムカウンタは、その行の開始命令のアドレスを指しています。また、カレント・プログラムカウンタは次の命令の参照ポイントとして便利です。カレント・プログラムアドレスを記憶して置か、計算するかわりに \$ というシンボルを使ってカレント・プログラムアドレスの値を使用することができます。

```
例)      JMP    $+ 6
        A      EQU    $
```

オペランドとしての 8080 オペコード

8080 オペコードは、8080 モードでのみ有効な 1 バイトのオペランドとなります。オペコードはアセンブル中にオペコードの 16 進の値で評価されます。

オペランドとして 8080 オペコードを使用するためには、8080 擬似命令をセットします。

オペコードの先頭 1 のバイトのみがオペランドとして有効です。オペランドとして用いるオペコードが 2 バイト以上を生成する命令の場合、かっこで囲んでオペコードの先頭 1 バイトを生成するようアセンブラに指示してください。

```
例) MVI      A,(JMP)
    ADI      (CPI)
    MVI      B,(RNZ)
    CPI      (INX H)
    ACI      (LXI B)
    MVI      C,MOV A,B
```

オペランド内に (かっこ内に) 2 バイト以上のコードが生成されるような命令が含まれると、——たとえば、(CPI 5)、(LXI B, LABEL1)、または (JMP LABEL2) などの場合はエラーとなります。

通常 1 バイトを生成するオペコードはかっこで囲まなくてもオペランドとして使用することができます。

3.4.2 演算子（オペレータ）

MSX・M-80のオペレータには算術オペレータおよび論理オペレータの2通りがあります。真偽判断をするオペレータ（すなわち論理オペレータ）は計算結果が0以外の数値のときに“真（True）”を、0の時に“偽（False）”を返します。次のオペレータは式中で 사용할 ことができます。

オペレータ定義

NUL

アーギュメント（パラメータ）がヌル（空白）の場合真を返します。NULに続く部分（同一行中）はNULのアーギュメントとして使用されます。

例) IF NUL <アーギュメント>

上の条件はアーギュメントの最初の文字がセミコロンまたはキャリッジリターン以外のものであった場合に偽となります。上の条件はIFBやIFNBの機能と同じですが、IFBの方が使用法が簡単です（3.6の条件アセンブル機能の項を参照してください）。

TYPE

TYPEオペレータはアーギュメントのモードと外部参照であるかそうでないかの情報を含むバイトを返します。TYPEのアーギュメントは任意の式（ストリング、数値、論理）です。その式が無効の場合TYPEはゼロを返します。返されるバイトは次のとおりです。

TYPE オペレータで得られる情報

2⁷.....外部参照ビット 1 = 式に外部参照を含む 0 = 式は内部参照

2⁶

2⁵.....定義ビット 1 = 式が内部で定義済 0 = 式が内部で未定義

2⁴

2³

2²

2¹..... } 式のモード 0 = 絶対モード 1 = プログラム相対モード

2⁰..... } 2 = データ相対モード 3 = COMMON 相対モード

- 下位2ビットはモードを表します。

0 絶対モード

1 プログラム相対モード

2 データ相対モード

3 COMMON 相対モード

- 上位ビット（80 H）は外部ビットです。この上位ビットがオンの場合式には外部参照が含まれます。また上位ビットがオフの場合式は内部参照となります。

- 式がモジュール内で定義されている場合は 20 H のビットがオンです。このビットは式が内部で定義されている場合はオンであり、式が定義されていないか外部参照の場合はオフです。全部のビットがオフの場合その式は無効です。
- TYPE は通常マクロ内部で使います。マクロ内で条件アセンブリを呼び出すときなどにアーギュメントのタイプをテストして判断することができます。

```
例)  FOO      MACRO      X
      LOCAL    Z
      Z        SET TYPE  X
      IF       Z...
```

TYPE は X のモードおよびタイプをテストします。X の評価によって IF Z.. で始まるコードのブロックをアセンブルする場合と、アセンブルしない場合とに分けることができます。

LOW

絶対 16 ビット値の下位 8 ビットを分離します。

HIGH

絶対 16 ビット値の上位 8 ビットを分離します。

*

乗算を行います。

/

除算を行います。

MOD

剰余の計算をします。左オペランドを右オペランドで割り、その残り（剰余）の値を返します。

SHR

右シフトを行います。SHR に続く整数の数だけ右へシフトが行われます。

SHL

左シフトを行います。SHL に続く整数の数だけ左へシフトが行われます。

— (負号)

負号を表わすオペレータです。このオペレータに続く整数は負の値であることを示します。

+

加算を行います。

-

左オペランドから右オペランドを減算したものを返します。

EQ

等号の条件判断。両オペランドが等しい場合には真を返します。

NE

不等号の条件判断。両オペランドが等しくない場合には真を返します。

LT

左オペランドが右オペランドより小さい場合には真を返します。

LE

左オペランドが右オペランドより小さいかあるいは等しい場合には真を返します。

GT

左オペランドが右オペランドより大きい場合に真を返します。

GE

左オペランドが右オペランドより大きいかあるいは等しい場合に真を返します。

NOT

- ①否定。オペランドが偽である場合、真を返します。オペランドが真である場合、偽を返します。
- ②オペランドの全ビットを反転した値を返します。

AND

- ①論理積。両オペランドが真の場合に真を返します。いずれかのオペランドが偽であるか、両オペランドが偽の場合に偽を返します。両オペランド共に絶対値でなければなりません。
- ②左オペランドと右オペランドの論理積を返します。

OR

- ①論理和。いずれかのオペランドが真であるか、両オペランドが真である場合に真を返します。両オペランド共に絶対値でなければなりません。
- ②左オペランドと右オペランドの論理和を返します。

XOR

①排他的論理和。いずれかのオペランドが真であり他方が偽である場合に真を返します。両オペランドが共に真であるか、両オペランドが共に偽である場合に偽を返します。両オペランドは共に絶対値でなければなりません。

②左オペランドと右オペランドの排他的論理和を返します。

オペレータの優先順位は以下のとおりです。

NUL, TYPE
 LOW, HIGH
 *,/, MOD, SHR, SHL
 負号
 +, -
 EQ, NE, LT, LE, GT, GE
 NOT
 AND
 OR, XOR

左よりも優先順位の高いオペレータを呼び出す副式が最初に計算されます。優先順位は優先したいと望む式の部分をカッコで囲むことによって変更することができます。

+, -, *, および/を除くすべてのオペレータは、1つ以上の空白を入れてオペランドと分離しなければなりません。

バイト分離オペレータ (HIGH および LOW) は、16 ビットの値の上位と下位 8 ビットを分離します。

3.5 アセンブラの機能

MSX・M-80 マクロアセンブラは単一機能を持つ擬似命令、マクロ機能、および条件アセンブル機能の3つの一般機能を備えています。

3.6 擬似命令

擬似命令は単一の機能を行うようアセンブラに指示します。これに対してマクロおよび条件付アセンブルは、1つ以上の擬似命令を呼び出すのでブロック擬似命令として考えることができます。

擬似命令は、次の5つのタイプに分類することができます。

- 命令セット選択
- データ定義／シンボル定義
- PC モード指定
- ファイル相対
- リスティング制御

命令セットの選択

MSX・M-80 ではアセンブルしようとするプログラムが Z 80 命令セットを使用するか、8080 命令セットを使用するかを選択することができます。

命令セットのデフォルトは 8080 です。Z 80 の命令セットを使用するプログラムでは、プログラムの先頭で Z 80 命令セットを使用することを擬似命令によって宣言しなければなりません。本マニュアルで取り扱う擬似命令は SET 擬似命令を除いてどちらの命令セットを選択しても使用できます。

命令セット選択擬似命令に続いて入力されるすべてのオペコードは、別の命令セット選択擬似命令に出会うまで指定した命令セットのオペコードとしてアセンブルされます。選択した命令セットにないオペコードを入力するとオブジェクショナブル・シンタックスエラー（文字 O Objectionable Syntax Error）となります。命令セット選択擬似命令は次のとおりです。

```
. Z 80
. 8080
```

. Z 80 (Z 80 モード選択)

機 能 Z 80 命令セットモードにします。

書 式 . Z 80

文 例 . Z 80

解 説

- Z 80 命令セットモードでアセンブルするよう MSX・M-80 に指示します。
- アーギュメントはありません。

. 8080 (8080 モード選択)

機 能 8080 命令セットモードにします。

書 式 . 8080

文 例 . 8080

解 説

- 8080 命令セットモードでアセンブルするよう MSX・M-80 に指示します。
- アーギュメントはありません。命令セット選択の擬似命令がない場合、デフォルトとして Z 80 命令セットを仮定します。

データ定義およびシンボル定義

データ定義およびシンボル定義擬似命令は SET 擬似命令を除いて、8080/Z80 のどちらの命令セットモードでもサポートされます。SET 擬似命令は Z80 モードでは使用できません。データ定義およびシンボル定義の擬似命令は次のとおりです。

DB/DEFB/DEFM
DC
DS/DEFS
DW/DEFW
EQU
EXT/EXTRN/EXTERNAL
BYTE EXT/BYTE EXTRN/BYTE EXTERNAL
ENTRY/GLOBAL/PUBLIC
SET/DEFL/ASET

DB/DEFB/DEFM (バイト定義)

機能 アーギュメントで指定したストリングまたは式の値を持つ1バイトずつのデータ領域を確保し、初期値をセットします。

書式 [<ラベル>] DB <式> [, <式>...]
 [<ラベル>] DEFB <式> [, <式>...]
 [<ラベル>] DB <ストリング> [, <ストリング>...]
 [<ラベル>] DEFM <ストリング> [, <ストリング>...]

文例 DB 'AB'
 DB 'AB' AND OFFH
 DB 'ABC'
 アセンブル結果は次のとおりです。
 0000' 41 42 DB 'AB'
 0002' 42 DB 'AB' AND OFFH
 0003' 41 42 43 DB 'ABC'

解説

- DEFB はアーギュメントとして式を指定する場合に使用します。また、DEFM はアーギュメントとしてストリングを指定する場合に使用します。
- アーギュメントで指定した値はカレント・ロケーションカウンタの位置から1バイトずつセットされます。
- <式> の値は2バイトで計算し、下位1バイトデータとして定義します。
- <式> の値は(±)0~225(1バイトで表すことができる値)でなければなりません。それ以外の場合には"A"エラーとなります。(式)の値の上位バイトのビットはすべて0、またはすべて1でなければなりません)
- 式中のストリングは3文字以下です。ストリングを複数個指定する場合はカンマ(,)で区切って続けるか、次の行に続けて書きます。8080/Z80のストリングは最上位ビットを0にした1バイトの値でアーギュメントの順序に従って用意されます。

DC (文字定義)

機能 アーギュメントで指定したストリングのデータ領域を確保して初期値をセットします。

書式 [**ラベル**] DC <ストリング>

文例 F00: DC "ABC"

アセンブル結果は次の通りです。

0000' 41 42 C3 F00: DC "ABC"

- 解説**
- アーギュメントで指定した<ストリング>がカレント・ロケーションカウンタの位置からの連続した領域に用意されます。
 - ストリング中の各文字は最上位ビットが0の1バイトで用意されますが、ストリングの最後の文字は最上位ビットが1になります。
 - アーギュメントがヌルストリングの場合はエラーとなります。

DS/DEFS (領域定義)

機能 メモリ領域を予約 (確保) します。

書式 [**ラベル**] DS **式** [, **値**]
 [**ラベル**] DEFS **式** [, **値**]

文例 DS 100 H ……100 H バイトのメモリ領域を予約します。
 DS 100 H, 2 ……100 H バイトのメモリ領域を予約し、各バイトの値を 2 にします。

解説

- DS/DEFS は **式** の値の大きさのメモリ領域を確保します。
- **値** は確保した領域の初期設定値です。**値** を省略すると領域の初期設定はされず、実行時のプログラム開始時の内容は保証されません。ただし、アセンブル時に MSX・M-80 のスイッチの /M を使用すると **値** の指定のないメモリ領域を自動的にゼロに初期設定をします。詳細は 4.5 のスイッチを参照ください。
- **式** 中にシンボルを使用する場合、シンボルは同一モジュール内で定義されていなければなりません。同一モジュール内で定義されていないと次のいずれかのエラーとなります。
 - a. 1 回目のパスで V エラー、2 回目のパスで U エラーが発生。
 - b. 2 回目のパスで U エラーが発生しない場合フェーズエラーが発生。
- 領域定義擬似命令は 1 回目のパスではコードを生成せず、2 回目のパスで領域の確保の処理をします。

DW/DEFW (ワード定義)

機 能 2 バイトのワードを確保し、初期値をセットします。

書 式 [**<ラベル>**] DW **<式>** [, **<式>**...]
[**<ラベル>**] DEFW **<式>** [, **<式>**...]

文 例 FOO: DW 1234H
アセンブル結果は次のとおり
0000' 1234 FOO: DW 1234H

- 解 説**
- カレント・ロケーションカウンタからの連続した領域に **<式>** の数だけ 2 バイトのワード領域を確保し、式の値をセットします。
 - **<式>** の値はメモリ上では下位バイト、上位バイトの順にセットされます。



- 生成されたコードはアセンブルリストでは上の例のように上位バイト、下位バイトの順に表示されます。

EQU

機 能 アーギュメントで指定した〈式〉の値に〈シンボル〉を割り当てます。

書 式 〈シンボル〉 EQU 〈式〉

文 例 BUF EQU 0F3H

解 説

- 〈シンボル〉は〈式〉の中で使用することができます。
- 〈シンボル〉の直後にはコロン（:）を付けません。
- すでに〈シンボル〉が〈式〉以外の値をもっている場合は M エラーとなります。あとでプログラム内で〈シンボル〉を再定義する場合には、EQU の代わりに SET または ASET 擬似命令を使用します。SET の項を参照してください。

EXT/EXTRN/EXTERNAL

BYTE EXT/BYTE EXTRN/BYTE EXTERNAL(エクスターナルシンボル)

機 能 アーギュメント中のシンボルが外部参照（他のモジュールで定義されているパブリックシンボルを参照）していることを宣言します。

書 式

```
EXT <シンボル> [, <シンボル>...]  
EXTRN <シンボル> [, <シンボル>...]  
EXTERNAL <シンボル> [, <シンボル>...]  
BYTE EXT <シンボル> [, <シンボル>...]  
BYTE EXTRN <シンボル> [, <シンボル>...]  
BYTE EXTERNAL <シンボル> [, <シンボル>...]
```

文 例

```
EXTRN          ITRAN          ; tranf init RTN
```

解 説

- アーギュメント中のシンボルが同一モジュール内に定義してあるシンボルの場合、M エラーとなります。
- EXT, EXTRN, EXTERNAL は 2 バイトの値として、また、BYTE EXT, BYTE EXTERNAL, BYTE EXTRN は 1 バイトの値として評価されます。
- EXTRN と EXT, BYTE EXTRN と BYTE EXT とはそれぞれ同じ働きをします。
- アーギュメント中のシンボルは、同一モジュール中で定義したシンボルと同じように使用することができます。
- エクスターナルシンボル擬似命令でエクスターナルシンボルの宣言をしない場合、シンボルの直後にポンド記号 2 つ (##) 付けて同一モジュール内で定義したシンボルと同じように使うことができます。

```
例)      EXTRN    SUB 1  
          CALL     SUB 1  
例)      CALL     SUB 1##
```

- エクスターナルシンボルは先頭 6 文字が外部参照可能（リンクロードに渡される）であり、以降の文字は MSX・M-80 の内部で切り捨てられます。
- MSX・M-80 はこの擬似命令についてのコードは生成しません。文例の EXTRN を含むモジュールでは、

```
CALL ITRAN
```

の CALL についてコードを生成するとき、ITRAN というシンボルの値をゼロ (0000)₁₆ で生成します。リンクロード時にロードされた他のモジュールの中から PUBLIC ITRAN を含むモジュールを検索し、ITRAN が定義された値（アドレス）に置き換わります。

ENTRY/GLOBAL/PUBLIC (パブリックシンボル)

- 機能** 同一モジュール内で定義しているシンボルを他のモジュールから参照できるように宣言します。
- 書式** ENTRY <シンボル> [, <シンボル>,...]
 GLOBAL <シンボル> [, <シンボル>,...]
 PUBLIC <シンボル> [, <シンボル>,...]
- 文例** PUBLIC LABEL1, LABEL2, DATA
- 解説**
- この擬似命令で宣言したパブリックシンボルは MSX・L-80 で一緒にリンクロードする他のモジュールで参照することができます。
 - ENTRY, GLOBAL, PUBLIC の機能は同じです。
 - シンボルはモジュール内で定義されていなければなりません。シンボルがモジュール内で定義されていない場合 U エラーになります。また、シンボル名が外部参照名またはコモンブロック名の場合 M エラーとなります。
 - パブリックシンボル名は先頭 6 文字が外部参照可能（リンクローダに渡される）で、以降の文字は MSX・M-80 の内部で切り捨てられます。
 - MSX・M-80 はこの擬似命令についてのコードは生成しません。次の例のモジュール 1 とモジュール 2 のリンクロード時に、モジュール 2 の ITRAN シンボルの値（アドレス）がモジュール 1 の CALL ステートメントの ITRAN の値として使用されます。
 - シンボルの直後にコロンを 2 つ (:) 付加すると、パブリックシンボル擬似命令で、パブリックシンボルの宣言をしなくても、パブリックシンボル擬似命令で宣言をしたのと同じように、他のモジュールから参照することができます。

サンプルプログラム

モジュール 1

EXTRN ITRAN

.
.
.

CALL ITRAN

モジュール 2

PUBLIC ITRAN

.
.
.

```

ITRAN:  LD      HL, PASSA
        ;STORE ADDR OF
        ;REG PASS AREA

```

SET/DEFL/ASET (セット)

機 能 アーギュメントで指定した〈式〉の値に〈シンボル〉を割り当てます。

書 式 〈シンボル〉 SET 〈式〉
 〈シンボル〉 DEFL 〈式〉
 〈シンボル〉 ASET 〈式〉

文 例 FOO ASET BAZ + 1000 H
 .
 .
 .
 FOO ASET 3000 H
 .
 .
 .
 FOO DEFL 6 CDEH

- 解 説**
- 〈シンボル〉は〈式〉の中で続けて使用することができます。また、〈シンボル〉の直後にはコロン(;)を付けません。式がエクスターナルシンボルを含む場合はエラーとなります。
 - SET 擬似命令は Z 80 のオペコードに SET 命令があるため、Z 80 命令セットモードでは使用できません。ASET と DEFL はどちらの命令セットモードでも使用が可能であり、機能は同じです。
 - セット擬似命令は EQU の代わりに〈シンボル〉の再定義をすることができます。いずれかのセット擬似命令で一旦定義した〈シンボル〉は、前に使用したセット擬似命令にかかわらずどのセット擬似命令でも再定義することができます (ただし Z 80 命令セットモードでは SET を使用できません。EQU の項を参照してください)。

PCモード

コードの生成はプログラムカウンタ（略してPC）と呼ばれるカレント・ロケーションカウンタのアドレスを基準として行います。PCは次のステートメントのコード生成アドレスを指しています。

MSX・M-80ではPCは絶対モード、データ相対モード、コード相対モード、またはCOMMON相対モードのいずれか1つのモードをもっています。PCモードの指定は以降のプログラムをそれぞれ絶対セグメント、コード相対セグメント、データ相対セグメント、COMMONセグメントの対応するセグメントとしてアセンブルします。プログラム中のシンボルや式のモードは、PCモードと同じモード属性になります。詳細については3.2.5を参照ください。PCモード擬似命令はPCモードを設定し、PCモード擬似命令以降のプログラムをどのモードのセグメントにアセンブルするかを決定します。PCモードの擬似命令は次のとおりです。

```
ASEG
CSEG
DSEG
COMMON
ORG
.PHASE/.DEPHASE
```

ASEG (絶対モード)

機能 PC モードを絶対モードにします。

書式 ASEG

文例 ASEG
ORG 103H

解説

- ASEG 以降のプログラムは絶対モードのセグメントにコード生成されます。
- ASEG にはアークギュメントはありません。
- ASEG はロケーションカウンタをメモリの絶対セグメント (実アドレス) にセットします。
ロケーションカウンタのデフォルト値は 0 であるため、ASEG を指定しただけではオペレーティングシステムの領域にプログラムを配置することになります。ASEG ステートメントの次に ORG ステートメントでロケーションカウンタの値を 103 H 以上に指定してください。

CSEG (コード相対モード)

機能 PC モードをコード相対モードにします。

書式 CSEG

文例 CSEG

解説

- CSEG 以降のプログラムはコード相対モードのセグメントにコード生成されます。コード相対セグメントはリロケートブルです。
- CSEG にはアーギュメントはありません。
- CSEG はロケーションカウンタをコード相対セグメントの先頭を 0 とするポイントに切り換えます。
- ロケーションカウンタの初期値は 0 であり、ORG ステートメントで変更しない限り生成コードの大きさだけ増加します。
- コード相対モードでは ORG ステートメントで絶対アドレスを指定することはできません。ORG ステートメントではリロケートブルな〈式〉を指定します。また〈式〉で数値を指定すると、ロケーションカウンタに〈式〉の値を加算します。

次の例はロケーションカウンタを 50 バイト大きくします。CSEG ステートメントの次に ORG ステートメントを置くことにより、コード相対セグメント中のコード生成アドレスを相対的に変えることができます。

例) CSEG

ORG 50

コード部分 A

DSEG

データ部分 B

CSEG

コード部分 C

- コード相対セグメントのメモリ配置位置は MSX・L-80 でリンクロードするときに /P スイッチを使って任意のアドレスに指定することができます。
- CSEG は MSX・M-80 のデフォルトのモードです。ASEG, DSEG, または COMMON 擬似命令を指定するまでプログラムをコード相対セグメントとしてアセンブルします。

DSEG (データ相対モード)

機 能 PC モードをデータ相対モードにします。

書 式 DSEG

文 例 DSEG

解 説

- DSEG 以降のプログラムはデータ相対モードのセグメントにコード生成されます。データ相対セグメントはリロケートブルです。
- DSEG にはアーギュメントはありません。
- DSEG はロケーションカウンタをデータ相対セグメントの先頭を 0 とするポインタに切り換えます。
- ロケーションカウンタの初期値は 0 です。ORG ステートメントで変更しない限り、生成コードの大きさだけ増加します。
- データ相対モードでは ORG ステートメントで絶対アドレスを指定することはできません。ORG ステートメントではリロケートブルな〈式〉を指定します。また〈式〉で数値を指定すると、ロケーションカウンタに〈式〉の値を加算します。
- データ相対セグメントのメモリ配置位置は MSX・L-80 でリンクロードするときに/D スイッチを使って任意のアドレスに指定することができます。

COMMON (COMMON 相対モード)

機能 COMMON 以降を COMMON 相対モードにします。

書式 COMMON /〈ブロック名〉/

文例

	COMMON	/DATA IN/
ANVIL	EQU	100 H
	DB	OFFH
	DW	1234 H
	DC	'FORGE'
	CSEG	
	.	
	.	
	.	

- 解説**
- COMMON 相対モードのセグメントは他のモジュールと共用できる共通データエリアです。
 - 〈ブロック名〉は6文字までです。7文字以上の〈ブロック名〉を記述すると、先頭6文字までが〈ブロック名〉として使用され、以降の文字は無効となります。
 - COMMON は同じ〈ブロック名〉を持つ COMMON ブロックごとに共通データエリアを作成します。〈ブロック名〉を省略しているか、スペースである場合、〈ブロック名〉がブランクの共通データエリアと仮定されます。
 - COMMON は、変数、配列、およびストリングを COMMON ストレージと呼ばれるストレージエリアに割り当て、異なるモジュール間で同一のストレージエリアを共用することができます。
 - COMMON 以降のステートメントは〈ブロック名〉の COMMON エリアとしてアセンブルされます。
 - COMMON エリアの大きさは別の PC モード擬似命令までの COMMON ブロック内の変数、配列、データの領域長の合計です。
 - MSX・L-80 でリンクロードしようとする複数モジュールに大きさの異なる同一名の COMMON ブロックがある場合、最も大きな COMMON ブロックを含むモジュール名を MSX・L-80 コマンドの先頭に指定し、最初にロードしなければなりません。MSX・L-80 の詳細は第5章を参照してください。
 - COMMON はロケーションカウンタを共通ブロックポインタに切り換えます。ロケーションは常に FORTRAN の COMMON ステートメントとの互換性を保つためにエリアの始めにあります。

ORG (セットオリジン)

機 能 ロケーションカウンタの値を変更します。

書 式 ORG <式>

文 例

- 1) DSEG
 ORG 50
- 2) ASEG
 ORG 800 H

解 説

- ORG は絶対モードではロケーションカウンタに<式>の値をセットします。MSX・M-80 は<式>の値のアドレスからコードを生成します。
- ORG はコード相対モード、データ相対モード、COMMON 相対モードではロケーションカウンタに <式> の値を加算します。MSX・M-80 はロケーションカウンタに <式> の値を加算したアドレスからコードを生成します。
- コード相対モード、データ相対モードでは <式> で絶対アドレスを指定することができません。コード相対セグメント、データ相対セグメントのロードアドレスは MSX・L-80 でリンクロード時にフレキシブルに決まります。しかし、MSX・L-80 のスイッチ/P: <アドレス> や/D <アドレス> を指定することにより、任意に指定したアドレスへロードすることができます。
- <式> に使用するシンボルは同一モジュール内で定義しなければなりません。また、<式> の値は絶対値またはロケーションカウンタで使用するエリアと同一のエリアになければなりません。
- 文例の 1) はデータ相対モードのロケーションカウンタをデータ相対セグメントの先頭+50 に設定します。データ相対セグメントの先頭 50 バイトには 0 が詰められます。/P スイッチの詳細については 5.5 を参照してください。
- 文例の 2) は絶対モードのロケーションカウンタを絶対アドレスの 800 H に設定します。2) の絶対セグメントは絶対アドレス 800 H からロードされます。

. PHASE / . DEPHASE (リロケート)

機能

. PHASE 以降, . DEPHASE までのプログラムを〈式〉で指定したエリアで実行します。

書式

. PHASE 〈式〉

・
・
・

. DEPHASE

文例

```

.PHASE      100 H
FOO:  CALL  BAZ
      JMP   ZOO
BAZ:   RET
      .DEPHASE
ZOO:   JMP   5

```

解説

- . PHASE / . DEPHASE は特定の絶対アドレスでプログラムの一部分を実行するために用います。
- . PHASE / . DEPHASE 間のプログラムの PC モードは絶対モードですが, アセンブル時には, . PHASE を指定する直前のプログラムと同じセグメントに連続して配置されます。
- . PHASE / . DEPHASE 間のプログラムは実行時に〈式〉で指定したアドレスへリロケートしてから実行してください。
- 〈式〉は絶対値です。
- 文例は次のようにアセンブルされます。

```

.PHASE      100 H
0100  CD  0106  FOO:  CALL  BAZ
0103  C 3  0007'  JMP   ZOO
0106  C 9          BAZ:  RET
      .DEPHASE
0007'  C 3  0005  ZOO:  JMP   5
                        END

```

ファイル関係

ファイル関係の擬似命令はプログラムに長いコメントを入れたり，モジュールに名前を付けたりします．ファイル関係の擬似命令は次のとおりです．

.COMMENT

END

INCLUDE/\$INCLUDE/MACLIB

NAME

.RADIX

.REQUEST

. COMMENT (コメント)

機能 . COMMENT 以降の〈区切り記号〉ではさまれた〈テキスト〉をコメントとみなします。

書式 . COMMENT 〈区切り記号〉〈テキスト〉〈区切り記号〉

文例 . COMMENT * any amount of text

entered here

.

.

* ; return to normal assembly

- 解説**
- . COMMENT のあとの最初の空白以外の文字を区切り文字として扱います。〈区切り記号〉以降、次の〈区切り記号〉までの〈テキスト〉がコメントブロックとなります。
 - コメント行のようにコメントの前にセミコロン(;)を置く必要はありません。〈テキスト〉は複数行にわたって書くことができます。
 - アセンブル中コメントブロックは無視され、コードの生成は行われません。

END (プログラムの終了)

機能 モジュールの終了を指定します。また、オプションでプログラムの実行開始点を指定します。

書式 END [<式>]

文例 1) END PROG1

2) END

- 解説**
- <式> はモジュールがメインプログラムであるときに、プログラムの開始点の指定するために使用します。他のプログラムから呼び出されるサブルーチンの場合、<式>を指定する必要はありません。
 - <式> は MSX・L-80 がプログラムの開始点としてロードすることができるシンボル、数値、またはその他のアーギュメントです。
 - <式> を指定すると MSX・L-80 は 100H にあるプログラム開始点へのジャンプ命令 (8080 JMP 命令) に <式> で指定したアドレスをセットします。
 - <式> を指定しない場合は MSX・L-80 にプログラム開始点が渡されません。リンクロード時にプログラム開始点を指定したモジュールがない場合、最初にロードするモジュールの先頭をプログラム開始点として指定します (<式> を指定しない場合、MSX・L-80 のスイッチ/G は無効です)。
 - プログラム開始点を持つモジュールを 2 つ以上リンクロードする場合は、MSX・L-80 スイッチ/G または /E を指定してください。
 - 高級言語プログラムではコンパイル時に自動的にプログラム開始点を設定します。高級言語で作成したモジュールを念み、かつアセンブリ言語で作成したモジュールから実行を開始したいプログラムの場合、高級言語モジュールの始めに CALL ステートメントなどを使って高級言語モジュールの実行に先立ちアセンブリ言語モジュールを呼び出します。
 - END ステートメントは MSX・L-80 にモジュールの終了を宣言します。ソースプログラムの最後に END ステートメントがない場合、%NO END 警告エラーメッセージが表示されます。

INCLUDE/\$INCLUDE/MACLIB (挿入)

機能

INCLUDE ステートメントの位置に他のソースファイルに含まれるステートメントを論理的に挿入します。

書式

```
INCLUDE <ファイル名>
$INCLUDE <ファイル名>
MACLIB <ファイル名>
```

文例

```
INCLUDE TEXT
```

解説

- INCLUDE, \$INCLUDE, MACLIB の機能は同じです。インクルード擬似命令を使用することにより、頻繁に使用する部分を繰り返しプログラミングする手間を省くことができます。
- <ファイル名> の形式は <デバイス指定>: <ファイル名>, <拡張子> です。ソースファイルの拡張子のデフォルトは .MAC です。デバイス指定のデフォルトはアセンブル時のデフォルトドライブです。デバイス指定および拡張子がデフォルトと同じである場合は省略することができます。デバイス指定は大文字で行なって下さい。
- <ファイル名> で指定するファイルをインクルードファイルと呼びます。インクルード擬似命令はインクルード擬似命令の位置にインクルードファイル中に含まれる全テキストを論理的に挿入し、アセンブルします。ソースファイルは変化しません。インクルードファイルの最後のステートメントをアセンブル後、インクルード擬似命令の次のステートメントのアセンブルをします。
- インクルードのネスト (入れ子) は許されません。インクルードのネストがあるとオブジェクトシヨナブル・シンタックスエラーとなります。
- <ファイル名> で指定したファイルがアセンブル時にみつからない場合、または拡張子が .MAC でない場合には V エラーとなり、インクルード擬似命令は無視されます。
- アセンブルリストには挿入したインクルードファイルのステートメントがプリントされます。また挿入したステートメントのオブジェクトコードとソースラインの間には、他のステートメントとの区別するために "C" を表示します。リストフォーマットの詳細についてはリスティング制御命令を参照してください。

NAME (モジュール名定義)

機 能 モジュール名を定義します。

書 式 NAME (' <モジュール名>')

文 例 NAME ('MODUL 1')

解 説

- <モジュール名> はかっこ () とシングルクォーテーションで囲みます。モジュール名は先頭 6 文字までが有効です。
- モジュール名は TITLE 擬似命令でも定義することができます。NAME および TITLE 擬似命令の両方を指定していない場合、モジュール名はソースファイルのファイル名が設定されます。

. RADIX (基数指定)

機能 定数の入力基数を指定します。

書式 .RADIX <式>

文例 .RADIX 2

解説

- 定数のデフォルトの入力基数は 10 進ですが、2 進～16 進までの範囲で入力基数を任意の底に変更することができます。
- <式> は 2～16 の 10 進数の定数です。
- .RADIX はアセンブルリストの基数は変更しません。また生成コードの値は 16 進基数です。
- .RADIX で指定した基数と異なる基数を用いる場合は 3, 4, 1 で説明している表記法を使ってください。

サンプル
プログラム

```

DEC :      DB      20
          .RADIX   2
BIN :      DB      00011110
          .RADIX   16
HEX :      DB      0CF
          .RADIX   8
OCT :      DB      73
          .RADIX   10
DECI :     DB      16
HEXA :     DB      0CH

```

アセンブル結果

```

0000' 14      DEC :      DB      20
0002          .RADIX   2
0001' 1E      BIN :      DB      00011110
0010          .RADIX   16
0002' CF      HEX :      DB      0CF
0008          .RADIX   8
0003' 3B      OCT :      DB      73
000A          .RADIX   10
0004' 10      DECI :     DB      16
0005' 0C      HEXA :     DB      0CH

```

. REQUEST (リンクロード要求)

- | | |
|-----|--|
| 機 能 | 外部シンボルに対応するシンボルを定義している外部モジュールのリンクロードを要求します。 |
| 書 式 | . REQUEST <ファイル名> [, <ファイル名>] |
| 文 例 | . REQUEST SUBR 1 |
| 解 説 | <ul style="list-style-type: none">● REQUEST はリンクロードに対する要求です。リンクローダはエクスターナルシンボル (EXTRN や <シンボル>: :) の参照を解決するために <ファイル名> で指定したモジュールを検索し、対応するシンボルがあれば追加モジュールとしてロードします。● <ファイル名> には、ドライブ指定、拡張子を指定しません。また、<ファイル名> の長さは 7 文字までです。● 文例ではリンクロード時にエクスターナルシンボルに対応する PUBLIC シンボルがロード済のモジュールにない場合、SUBR 1 を検索します。 |

リスティング擬似命令

リスティング擬似命令はアセンブリリストのプリントに関する擬似命令です。

フォーマット制御

アセンブリリストのページブレイク、タイトル、およびサブタイトルを指示します。フォーマット制御擬似命令は次のとおりです。

```
PAGE/$EJECT
TITLE
SUBTTL/$TITLE
```

一般リスティング制御

アセンブリリストの出力／抑止を指定します。特に指定をしなくてもプログラムの先頭で、LIST を指定したものと同じとみなします。一般リスティング制御擬似命令は次のとおりです。

```
.LIST
.XLIST
```

条件リスティング制御

偽条件ブロック内に含まれるステートメントのアセンブリリストの出力／抑止を指定します。

4.5 の /X スイッチに関する記述も参照してください。条件リスティング制御擬似命令は次のとおりです。

```
.SFCOND
.LFCOND
.TFCOND
```

マクロ拡張リスティング制御

拡張リスティング擬似命令はマクロおよび REPT, IRP, IRPC で展開するステートメントのリスティングを制御します。拡張リスティング制御擬似命令はマクロまたはリピートブロック内のステートメントに対して有効です。指定がない場合は .XALL を指定しているものとみなします。マクロ拡張リスティング制御擬似命令は次のとおりです。

```
.LALL
.SALL
.XALL
```

CREF 情報出力制御擬似命令

CREF 情報出力制御擬似命令はクロスリファレンス情報をクロスリファレンス・ファイルへ出力するか抑止するかを指定します。本擬似命令は MSX・M-80 スイッチ /C を指定しないかぎり効果はありません。

CREF 情報出力制御擬似命令はソースプログラムの任意の位置に指定し、プログラムの必要な部分についてのクロスリファレンス情報を出力することができます。CREF 情報出力制御擬似命令は次のとおりです。

```
.CREF
.XCREF
```

PAGE/\$EJECT (フォームフィード)

機能 アセンブルリストの改ページを行います。

書式 PAGE <式>
\$EJECT

文例 \$EJECT 58

- 解説**
- 改ページを行いフォームフィード擬似命令以降のステートメントを次ページからプリントします。MSX・M-80 はページの終りのリスティングファイル内にフォームフィード文字を置きます。
 - <式> は改ページ後の 1 ページに出力する行数を指定します。<式> は 10～255 の数値です。デフォルトの行数は 50 行 / 1 ページです。

TITLE (タイトル)

機能 アセンブルリストにプリントするタイトルを指定します。

書式 TITLE <テキスト>

文例 TITLE PROG1

- 解説**
- TITLE はアセンブルリストの各ページの最初の行にプリントするタイトルを<テキスト>で指定します。1 つのモジュールに 2 つ以上の TITLE を指定すると Q エラーとなります。
 - 同じモジュール内に NAME 擬似命令を使用していない場合、<テキスト>の先頭 6 文字をモジュール名として設定します。TITLE、NAME 擬似命令がない場合、ソースファイル名をモジュール名として設定します。
 - 文例はアセンブルリストの各ページの最初の行に PROG1 というタイトルをプリントし、NAME 擬似命令がモジュール内になればモジュール名を PROG1 に設定します。

SUBTTL/\$TITLE (サブタイトル)

機 能 アセンブリリストにプリントするサブタイトルを指定します。

書 式 SUBTTL <テキスト>
\$TITLE ('<テキスト>')

文 例 SUBTTL SPECIAL I/O ROUTINE
.
.
.
SUBTTL
.
.
.

- 解 説**
- SUBTTL はアセンブリリストの各ページのタイトルの次の行にプリントするサブタイトルを指定します。
 - <テキスト> の 60 文字以降は無視します。1 つのモジュールに任意の数の SUBTTL を指定することができます。
- 既に SUBTTL を指定している場合、前の SUBTTL でセットした <テキスト> は最新の SUBTTL の <テキスト> で置き換わります。既に指定しているサブタイトルのプリントを解除する場合は <テキスト> の指定を省略してください。
- 文例の最初の SUBTTL により、サブタイトル SPECIAL I/O ROUTINE がアセンブリリストの各ページの先頭にプリントされます。2 番目の SUBTTL はサブタイトルのプリントを解除します (空白のサブタイトルがプリントされます)。

. LIST (リスト)

機 能 . LIST 以降のステートメントについてアセンブルリストを出力します。

書 式 . LIST

文 例

```

      .
      .
      .
      .XLIST      ; listing suspended here
      .
      .
      .
      .LIST      ; listing resumes here
```

- 解 説**
- .LIST はアセンブル開始時のデフォルト状態です。
 - MSX・M-80 コマンドでリスティングファイルを指定した場合に、指定したファイルにリストを出力します。

. XLIST (リスト抑止)

機能

. XLIST 以降のステートメントのアセンブルリストの出力を抑止します。

書式

. XLIST

文例

```

.
.
.
. XLIST ; listing suspended here
.
.
.
. LIST ; listing resumes here

```

解説

- アセンブル開始時のデフォルトは . LIST 状態です。 . XLIST を指定すると、ソースステートメントおよび生成したオブジェクトコードはプリントされません。 . XLIST は指定するまでのステートメントに対して有効です。
- . XLIST を指定すると . LIST 擬似命令以外のリスティング擬似命令のステートメントを無視します。したがって . LIST 以外のリスティング擬似命令があっても無効です。

. PRINTX (コンソールメッセージ表示)

機 能 アセンブル中、〈テキスト〉をコンソールに表示します。

書 式 . PRINTX 〈区切り記号〉〈テキスト〉〈区切り記号〉

文 例

```
1) . PRINTX * Assembly half done *  
2) IF1  
    . PRINTX * Pass 1 done * ; pass 1 message only  
    ENDIF  
    IF2  
    . PRINTX * Pass 2 done * ; Pass 2 message only  
    ENDIF
```

解 説

- . PRINTX は〈区切り記号〉にはさまれた〈テキスト〉をアセンブル時にコンソールへ表示します。
- . PRINTX 以降の空白以外の最初の文字を〈区切り記号〉とみなします。また、〈区切り記号〉から次の〈区切り記号〉までが〈テキスト〉です。
- . PRINTX は長時間かかる場合にアセンブルの進行状況を表示したり、アセンブルスイッチとして条件値を表示するときに役立ちます。
- . PRINTX はパス 1、パス 2 の両パスでそれぞれ〈テキスト〉をコンソールへ表示します。どちらかのパスでのみ表示をしたい場合、IF1、または IF2 擬似命令を使用してください。IF1 および IF2 については条件擬似命令を参照してください。

. SFCOND （偽条件抑止）

機 能 偽条件ブロックのアセンブルリスト出力を抑止します。

書 式 . SFCOND

文 例 . SFCOND

解 説

- .SFCOND は MSX・M-80 のスイッチ／X に関係なく、以後の偽条件ブロックのアセンブルリストの出力を抑止します。
- ／X スイッチを指定しない場合、偽条件ブロックに対するリスティング制御擬似命令がなければ偽条件ブロックのアセンブルリストが出力されます。また、／X スイッチを指定すると偽条件ブロックのアセンブルリストは出力されません。

. LFCOND （偽条件出力）

機 能 偽条件ブロックのアセンブルリストを出力します。

書 式 . LFCOND

文 例 . LFCOND

解 説

- .LFCOND は MSX・M-80 のスイッチ／X に関係なく以後の偽条件のブロックのアセンブルリストが出力されます。
- ／X スイッチを指定しない場合、偽条件ブロックに対するリスティング制御擬似命令がなければ偽条件ブロックのアセンブルリストは出力されます。また、／X スイッチを指定すると偽条件ブロックのアセンブルリストは出力されません。

. TFCOND (トグルリスティング偽条件)

機 能 /X スイッチの有無により偽条件ブロックのアセンブルリストの出力を決定する。また、. TFCOND の指定をするたびに出力／抑止を反転する。

書 式 . TFCOND

文 例 . TFCOND

解 説 ●偽条件ブロックのリスティング制御が未定の部分は、2つの. TFCOND で囲んでおく事により、次のような効果を得ることができます。

- a. /X を指定する時、奇数番目の. TFCOND とその次の. TFCOND までの間の偽条件ブロックのアセンブルリストは出力され、偶数番目の. TFCOND とその次の. TFCOND までの間の偽条件ブロックの出力は抑止されます。
- b. /X を指定しない時、奇数番目の. TFCOND とその次の. TFCOND までの間の偽条件ブロックのアセンブルリストの出力は抑止され、偶数番目の. TFCOND とその次の. TFCOND までの間の偽条件ブロックは出力されます。

●/X スイッチを指定しない場合、偽条件ブロックに対するリスティング制御擬似命令がなければ偽条件ブロックのアセンブルリストは出力されます。

また、/X スイッチを指定する場合、偽条件ブロックに対するリスティング制御擬似命令がなければ偽条件ブロックのアセンブルリストは出力されません。

. LALL

- | | |
|------------|---|
| 機 能 | マクロおよびリピートブロックの展開部分をアセンブルリストに出力します。 |
| 書 式 | . LALL |
| 文 例 | . LALL |
| 解 説 | ●オブジェクトコードを生成するステートメントか否かにかかわらず、マクロおよびリピートブロックの展開部分のすべてのステートメントをアセンブルリストに出力します。 |

. XALL

- | | |
|------------|---|
| 機 能 | マクロおよびリピートブロックの展開部分のうちオブジェクトコードを生成するステートメントをアセンブルリストに出力します。 |
| 書 式 | . XALL |
| 文 例 | . XALL |
| 解 説 | ●拡張リスティング制御擬似命令の指定がない場合は . XALL を指定しているものとみなします。 |

. SALL

- | | |
|------------|---|
| 機 能 | マクロおよびリピートブロックの展開部分のアセンブルリストへの出力を抑止します。 |
| 書 式 | . SALL |
| 文 例 | . SALL |

. CREF (クロスリファレンス作成)

機 能 クロスリファレンス・ファイルを作成します。

書 式 . CREF

文 例 . CREF

- 解 説**
- . CREF 以降. XCREF の指定をするまでクロスリファレンス・ファイルが作成されます。
 - . CREF はデフォルトの状態ですので、. XCREF 擬似命令を使用した後でクロスリファレンス・ファイルの作成を再開するときに使用してください。
 - . CREF の機能は MSX・M-80 のコマンドに /C スイッチを指定したときに有効です。

. XCREF (クロスリファレンス抑止)

機 能 クロスリファレンス・ファイルの作成を抑止します。

書 式 . XCREF

文 例 . XCREF

- 解 説**
- . XCREF は. CREF 擬似命令の指定を解除します.. XCREF 以降. CREF の指定をするまでの部分のクロスリファレンス・ファイルは作成されません.. XCREF を使用することによってクロスリファレンス情報の作成を指定した部分だけ抑止することができます。
 - . CREF と .XCREF は MSX・M-80 のコマンドに /C スイッチを指定したときに有効です。このため通常のリスティングファイル (クロスリファレンス・ファイルではない) を出力させる場合には、MSX・M-80 のコマンドに /C スイッチを指定しないだけでよく、.XCREF を使う必要はありません。

3.7 マクロ機能

マクロ機能を使用して、プログラム中頻繁にコーディングする部分をあらかじめマクロとして定義し、プログラム中の任意の場所で何回でも呼び出して展開することができます。マクロ機能により、同じコーディングを繰り返す必要がなくなります。

マクロ機能には反復擬似命令 (REPT, IRP, IRPC) を含みます。また、マクロのネスト (マクロの中で別のマクロを呼び出し、展開すること) の深さはメモリ領域の大きさでのみ制限されます。

マクロの定義

マクロ定義は MACRO ステートメントで始め、ENDM で終了します。マクロ定義に対するオブジェクトコードは生成されません。マクロ定義後マクロを呼び出すと、呼び出した位置にマクロの展開を行い、展開ステートメントのオブジェクトコードが生成されます。

MACRO～ENDM (マクロ定義)

機 能 マクロブロックを定義します。

書 式 <マクロ名> MACRO [<ダミー> [, <ダミー>,...]]

 .
 .
 .
 ENDM

- 解 説**
- MACRO ステートメント以降、ENDM までのステートメントをマクロとして定義します。
 - <マクロ名> は定義するマクロの名前です。<マクロ名> の長さは任意ですが、先頭 16 文字が有効です。マクロを定義した後 (ENDM ステートメント以降)、マクロは <マクロ名> をオペレーションフィールドに指定して呼び出し展開することができます。
 <マクロ名> はシンボルの規則に従います。
 - <ダミー> はマクロ定義のブロック内で使用している <ダミー> がマクロ呼び出し時の <パラメータ> で置き換わります。
 <ダミー> は 32 文字までの文字列です。また、<ダミー> の数は一行に入るまでいくつでも指定できます。<ダミー> と <ダミー> との間はカンマ (,) で区切ります。

**サンプル
プログラム**

```
MOVE            MACRO    X, Y
                 MOV     HL, Y
                 MOV     X, HL
                 ENDM
```

呼び出し例

```
                 MOVE     AREA2, AREA1
```

アセンブル結果

```
                 MOVE
+                MOV     HL, AREA1
+                MOV     AREA2, HL
```

マクロの呼び出し

マクロの呼び出しはマクロ定義後プログラムの任意の場所で行うことができます。マクロの呼び出しの後にマクロ定義をするとエラーとなります。呼び出しの形式は次のとおりです。

〈マクロ名〉 [〈パラメータ〉[, 〈パラメータ〉,...]]

- 〈マクロ名〉はMSX・M ステートメントで定義した〈マクロ名〉です。
- 〈パラメータ〉はMSX・M ステートメントの〈ダミー〉と左から順番に対応しています。
- 〈パラメータ〉の数は任意ですが、〈パラメータ〉の並びは一行以内に収まらなくてはなりません。また、〈パラメータ〉と〈パラメータ〉との間はカンマ(,)で区切ります。
- カンマで区切った〈パラメータ〉を山形かっこ(〈 〉)で囲むと、山形かっこ内のすべての項目を単一の〈パラメータ〉として扱います。

FOO 1, 2, 3, 4, 5 …… 5つのパラメータとして扱います。

FOO 〈1, 2, 3, 4, 5〉 …… 1つのパラメータとして扱います。

- 〈パラメータ〉の数とMACRO ステートメントの〈ダミー〉の数は一致しなくても構いません。〈パラメータ〉の数が〈ダミー〉の数より多い場合、余った〈パラメータ〉は無視されます。また〈パラメータ〉の数が〈ダミー〉の数より少ない場合、不足している〈パラメータ〉を空白とみなします。

サンプル
プログラム

```
EXCHNG  MACRO  X, Y
          PUSH  X
          PUSH  Y
          POP   X
          POP   Y
        ENDM
```

呼び出し例

```
LDA      2FH
MOV      HL, A
LDA      3FH
MOV      DE, A
EXCHNG   HL, DE
```

アセンブル結果

0000'	3A002F		LDA	2FH
0003'	67		MOV	HL, A
0004'	3A003F		LDA	3FH
0007	57		MOV	DE, A
			EXCHNG	HL, DE
0008'	E5	+	PUSH	HL
0009'	D5	+	PUSH	DE
000A'	E1	+	POP	HL
000B'	D1	+	POP	DE

反復擬似命令

反復擬似命令は反復ブロックを反復擬似命令の位置に指定した回数だけくり返し展開します。

反復ブロックの展開は REPT, IRP, または IRPC ステートメントで始め, ENDM, または EXITM で終了します。マクロと反復擬似命令との違いは次のとおりです。

- マクロはマクロブロックをあらかじめ定義しておき, あとで必要な場所で何回でも呼び出して展開することができます。反復擬似命令は反復ブロックを反復擬似命令の位置にくり返し展開します。他の場所から呼び出すことはできません。
- マクロではマクロ呼び出し時にパラメータを指定することによって, 展開するマクロを変更することができます。

REPT~ENDM (反復)

機能 反復ブロックを指定した数だけくり返し展開します。

書式 REPT <式>

.

.

.

ENDM

解説

- REPT と ENDM ステートメント間の反復ブロックを <式> の回数だけくり返し展開します。
- <式> は 16 ビットの符号なし整数として評価されます (<式> の値は 1~65535)。
- <式> がエクスターナルシンボルであったり定義していない項目である場合にはエラーとなります。

**サンプル
プログラム**

```

0000          X      SET      0
                                REPT    10    ; generates DB1-DB10
                                X
                                SET      X+1
                                DB        X
                                ENDM
0000 01      +      DB        X
0001 02      +      DB        X
0002' 03      +      DB        X
0003' 04      +      DB        X
0004' 05      +      DB        X
0005' 06      +      DB        X
0006' 07      +      DB        X
0007' 08      +      DB        X
0008' 09      +      DB        X
0009' 0A      +      DB        X
                                END

```


IRP～ENDM (反復)

機能 反復ブロックを指定したパラメータに従ってくり返し展開します。

書式 IRP <ダミー>, <<パラメータ> [, <パラメータ>...]>

・
・
・

ENDM

- 解説**
- IRP と ENDM ステートメント間の反復ブロックを <パラメータ> の数だけくり返し展開します。また、展開中に反復ブロックに含まれる <ダミー> を展開回数に従って左側からの <パラメータ> で順次置き換えます。
 - <パラメータ> は山形かっこで囲みます。
 - <パラメータ> がない場合 (< >), 反復ブロックは <ダミー> を取り除いて1回だけ展開します。

**サンプル
プログラム**

			IRP	X, <1, 2, 3, 4, 5, 6, 7, 8, 9, 10>
			DB	X
			ENDM	
0000'	01	+	DB	1
0001'	02	+	DB	2
0002'	03	+	DB	3
0003'	04	+	DB	4
0004'	05	+	DB	5
0005'	06	+	DB	6
0006'	07	+	DB	7
0007'	08	+	DB	8
0008'	09	+	DB	9
0009'	0A	+	DB	10

前の例を MACRO 定義内で使用する場合

			FOO	MACRO	X
				IRP	Y, <X>
				DB	Y
				ENDM	
				ENDM	
			FOO		<1, 2, 3, 4, 5, 6, 7, 8, 9, 10>
0000'	01	+	DB		1
0001'	02	+	DB		2
0002'	03	+	DB		3
0003'	04	+	DB		4
0004'	05	+	DB		5
0005'	06	+	DB		6
0006'	07	+	DB		7
0007'	08	+	DB		8
0008'	09	+	DB		9
0009'	0A	+	DB		10

マクロ呼び出しの時の〈パラメータ〉は角かっこで囲んでいるので単一のパラメータとして扱われます。角かっこはマクロの展開時に取り除かれ、IRP のパラメータ内の X と置き換わります。

IRPC～ENDM (反復)

機能 反復ブロックを指定した文字列に従ってくり返し展開します。

書式 IRPC <ダミー>, <文字列>

・
・
・

ENDM

解説 ● IRPC と ENDM ステートメント間の反復ブロックを<文字列>の文字数だけくり返し展開します。また、反復ブロックに含まれる<ダミー>は展開中に展開回数に従って左側から<文字列>中の文字で順次置き換えられます。

**サンプル
プログラム**

			IRPC	X, 0123456789
			DB	X
			ENDM	
0000'	00	+	DB	0
0001'	01	+	DB	1
0002'	02	+	DB	2
0003'	03	+	DB	3
0004'	04	+	DB	4
0005'	05	+	DB	5
0006'	06	+	DB	6
0007'	07	+	DB	7
0008'	08	+	DB	8
0009'	09	+	DB	9

EXITM

機 能 マクロや反復ブロックの展開を終了します。

書 式 EXITM

解 説

- EXITM を条件擬似命令と組み合わせて使用することにより、マクロや反復ブロックの展開を展開時の条件によって途中で終了することができます。
- EXITM をアセンブルするとマクロや反復ブロックの展開を中止し、残りの展開についてのオブジェクトコード生成を行いません。
- EXITM を含むブロックが他のブロックの中の子になっている場合、外側のブロックの展開は続行します。

サ ン プ ル プ ロ グ ラ ム	FOO	MACRO	X
	Y	SET	O
		REPT	X
	Y	SET	Y + 1
		IFE	Y - OFFH; test Y
		EXITM	; if true, exit REPT
		ENDIF	
		DB	Y
		ENDM	

FOO マクロは呼び出しステートメントで指定した数だけバイト定義を展開します。定義内容は '01' からの昇順です。呼び出し時に 255 より大きい数を指定しても、255 以降の展開はしません。

LOCAL (マクロシンボル)

機能 マクロ展開時にマクロ定義中の〈ダミー〉を..0000 から..FFFF の重複しないシンボルに置き換えて展開するよう指示します。

書式 LOCAL 〈ダミー〉 [, 〈ダミー〉 ...]

文例 LOCAL A, B, C

- 解説**
- LOCAL はマクロ定義中でのみ使用します。
 - マクロ定義中に LOCAL で指定した〈ダミー〉をシンボルとして使用しているステートメントがあれば、モジュール内で重複しない..0000 から..FFFF のシンボルに置き換えて展開されます。
 - LOCAL ステートメントはマクロ定義の最初に指定しなければなりません (MACRO ステートメントの次のステートメント)。
 - LOCAL ステートメントはシンボルを自動的に作成するので、マクロ内でのラベルの定義、参照をするときなどに便利です。
 - LOCAL ステートメントは..0000 から..FFFF までのシンボルを定義するので、プログラミング時に..0000 から..FFFF までのシンボルを定義しないようにしてください。

サンプル
プログラム

	FOO	MACRO	NUM,Y
		LOCAL	A,B,C,D,E
	A :	DB	7
	B :	DB	8
	C :	DB	Y
	D :	DB	Y + 1
	E :	DW	NUM + 1
		JMP	A
		ENDM	
		FOO	0C00H,0BEH
0000'	07 + ..0000 :	DB	7
0001'	08 + ..0001 :	DB	8
0002'	BE + ..0002 :	DB	0BEH
0003'	BF + ..0003 :	DB	0BEH + 1
0004'	0C 01 + ..0004 :	DW	0C00H + 1
0006'	C3 0000' +	JMP	..0000
		END	

特殊なマクロ演算子

マクロ定義中では次のような特殊な演算子を使うことができます。

- & ……テキストやシンボルの連結
- ; ; ……展開が行われないコメント
- ! ……感嘆符の次のキャラクタを文字通りアーギュメントして扱う。
- % ……式を現在指定されている基数で表わす数に変換する。

&

機能

テキストまたはシンボルを連結します。

文例

```
1) PROC &X EQU $
2)          MVI B, '&X'
```

解説

- &はマクロ呼び出しステートメントでは使用できません。
- 引用符で囲まれたストリング内でダミーパラメータを使う場合、また、シンボルの一部としてダミーパラメータを使う場合（文字列とダミーを連結する場合）にダミーの直前に&を置きます。&がないとき、展開時にダミーパラメータとの置き換えを行いません。

サンプルプログラム

```
ERRGEN      MACRO  X
ERROR&X:    PUSH   B
            MVI    B, '&X'
            JMP    ERROR&X
            ENDM
```

呼び出し例

```
ERRGEN      A
ERROR&A:    PUSH   B
            MVI    B, 'A'
            JMP    ERROR&A
```


;;

機能 展開を行わないコメント。

文例 MVI B, '&X' ; ; COMMENT

解説

- マクロ定義中、(;;)以降を展開しないコメントとして扱います。LALL 擬似命令の指定後でもマクロ展開のリストにはプリントされません。
- セミコロン1つ(;)のコメントはマクロ展開のリストにプリントされます。

サンプルプログラム

```
DBGEN      MACRO    X
;; DB      GENERATION
Y          SET      O
           REPT     X
Y          SET      X + 1
; * * * * *
           DB       X
           ENDM
           ENDM

呼び出し例
           DBGEN    3
; * * * * *
           DB       Y
; * * * * *
           DB       Y
; * * * * *
           DB       Y
```

!

機 能 アーギュメントやパラメータ中の感嘆符の次のキャラクタを文字通りアーギュメントとして扱います。

文 例 IRP X, <! ; , ! , ! , ! 4>

解 説 ● ! ; は < ; > と同じです。

サンプルプログラム

```

IRP      X, <! ; , ! , ! , ! 4>
DB       '&X'
ENDM
DB       ' ; '
DB       ' , '
DB       ' , '
DB       ' 4 '

```

%

機能

マクロのパラメータとして指定した〈式〉を .RADIX 擬似命令でセットした基数に従って表わす。

解説

- %は直後に続く式を展開時の基数（.RADIX で指定した基数、.RADIX の指定がなければ 10 進数）で表わす数に変換します。マクロ展開時には変換後の数がダミーと置き換わります。
- %はマクロ呼び出しステートメントのパラメータにのみ使用できます。
- %を使用する事により、パラメータとして指定したテキストをダミーと置き換えるだけでなく、パラメータに値を持つことができます。
- %に続く式は、DS 領域定義擬似命令の式の規則に従います。式は絶対値で評価できなければなりません（リロケータブルでない式）。

サンプルプログラム

```

PRINTE      MACRO      MSG, N
              .PRINTX   * MSG, N *
              ENDM
SYM1        EQU        100
SYM2        EQU        200
PRINTE      <SYM1 + SYM2 =>, % (SYM1 + SYM2)
+           .PRINTX   * SYM1 + SYM2 = 300 *

```

% (SYM1+SYM2) は式の値を計算後、ダミー-Nと置き換わります。% (SYM1+SYM2) の代わりに (SYM1+SYM2) を指定すると次のステートメントが展開されます。

```
.PRINTX * SYM1 + SYM2 = (SYM1 + SYM2) *
```

3.8 条件擬似命令

条件擬似命令はアセンブル時の特定の条件をテストし、その結果に応じてコードブロックを生成するかどうかを決めます。条件擬似命令の形式は次のとおりです。

```

IFxxxx  [アーギュメント]      COND  [アーギュメント]
.
.
.
[ELSE
.
. ]
ENDIF
ENDC

```

- IFxxxx は条件を終了するために、ENDIF と対応していなければなりません。また、COND は条件を終了するために、ENDC と対応していなければなりません。IFxxxx や COND に対応する ENDIF や ENDC がない場合、パス 1、パス 2 の終了時に "Unterminated condition" のメッセージが出力されます。また、対応する IFxxxx がない ENDIF、または対応する COND がない ENDC の場合、C エラーとなります。
- アセンブラは条件ステートメントの真偽を評価します。
- 評価結果が真である場合、ELSE ステートメントがあれば IFxxxx、COND から ELSE までのブロックをアセンブルし、ELSE 文以降 ENDIF、ENDC までのブロックを無視します。また、ELSE ステートメントがない場合、IFxxxx、COND から ENDIF、ENDC までのブロックをアセンブルします。
- 評価結果が偽である場合、ELSE ステートメントがあれば、IFxxxx、COND から ELSE までのブロックを無視し、ELSE 以降 ENDIF、ENDC までのブロックをアセンブルします。また、ELSE ステートメントがない場合、IFxxxx、COND から ENDIF、ENDC までのブロックを無視します。
- 条件ステートメントの真偽の評価については次のページに述べます。
- 条件は 255 レベルまでネスト（入れ子）することができます。
- 条件ステートメント中のアーギュメントはパス 1 で解決できるものでなければなりません。IF、IFT、COND、および IFF、IFE の〈式〉はすでに定義された値を用い、かつ〈式〉の値が絶対値でなくてはなりません。また、IFDEF、または IFNDEF のアーギュメント中のシンボルが IFDEF、または IFNDEF 以降で定義されている場合、パス 1 では未定義となりますが、パス 2 で定義済みとなります。
- 条件擬似命令では偽の条件が成立する場合、真の条件の場合とは別のコードを生成するために ELSE ステートメントを使うことができます。
- ELSE は IFxxxx または COND に対して 1 つだけ使用することができます。2 つ以上の ELSE をもつ条件、または条件がない ELSE は C エラーを引き起こします。

IFxxxx~ELSE~ENDIF/COND~ELSE~ENDC (条件)

書 式 IF <式>
IFT <式>
COND <式>

解 説 ●<式>がゼロ以外の値に評価された場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。

書 式 IFE <式>
IFF <式>

解 説 ●<式>が0と評価された場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。

書 式 IF1

解 説 ●アセンブルがパス1を実行中の場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。IF1はアーギュメントとしての式をもちません。

書 式 IF2

解 説 ●アセンブラがパス2を実行中の場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。IF2はアーギュメントとして式をもちません。

書 式 IFDEF <シンボル>

解 説 ●<シンボル>が定義されているか、または外部に宣言されている場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。

書 式 IFNDEF <シンボル>

解 説 ●<シンボル>が定義されていないか、または外部に宣言されていない場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。

書式 IFB <アーギュメント>**解説**

- <アーギュメント> は山形カッコで囲みます。
- <アーギュメント> がブランク (何も与えられていない) または空白 (2つの山形カッコ < > の中には何も入っていない) の場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。
- IFB (および IFNB) は、通常マクロブロックの内部で使用されます。IFB 擬似命令に続く式はたいいていダミーシンボルです。マクロが呼び出されると、ダミーはマクロ呼び出しによって渡されるパラメータに置き換えられます。マクロ呼び出しが IFB に続くダミーを置き換えるためのパラメータを指定しない場合、式はブランクとなりそのブロックはアセンブルされます。

書式 IFNB <アーギュメント>**解説**

- <アーギュメント> は山形カッコで囲みます。
- <アーギュメント> がブランクでない場合、条件は真とみなされ条件ブロック内のステートメントがアセンブルされます。
- IFNB (および IFB) は通常マクロブロックの内部で使用されます。IFNB 擬似命令に続く式はたいいていダミーシンボルです。マクロが呼び出されるとダミーはマクロ呼び出しによって渡されるパラメータに置き換えられます。マクロ呼び出しが IFNB に続くダミーを置き換えるためのパラメータを指定しない場合、式はブランクとはならずそのブロックはアセンブルされます。

書式 IFIDN <アーギュメント 1>, <アーギュメント 2>**解説**

- <アーギュメント 1> および <アーギュメント 2> は山形カッコで囲みます。
- ストリング <アーギュメント 1> がストリング <アーギュメント 2> と同一である場合に条件は真とみなされ、条件ブロック内のステートメントがアセンブルされます。
- IFIDN (および IFDIF) は通常マクロブロックの内部で使用されます。IFIDN ディレクティブに続く式はたいいていダミーシンボルです。マクロが呼び出されるとダミーはマクロ呼び出しによって渡されるパラメータに置き換えられます。マクロ呼び出しがダミーを置き換えるための2つのパラメータに同じものを指定すると、そのブロックはアセンブルされます。

書 式

IFDIF <アークギュメント 1>, <アークギュメント 2>

解 説

- <アークギュメント 1> および <アークギュメント 2> は、山形カッコで囲みます。
- ストリング <アークギュメント 1> がストリング <アークギュメント 2> と異なっている場合に条件は真とみなされ、条件ブロック内のステートメントがアセンブルされます。
- IFDIF (および IFIDN) は通常マクロブロック内部で使用されます。IFDIF に続く式は、たいていダミーシンボルです。マクロが呼び出されるとダミーはマクロ呼び出しによって渡されるパラメータに置き換えられます。マクロ呼び出しがダミーを置き換えるための 2 つのパラメータとして異なるものを指定した場合に、ブロックがアセンブルされます。



第4章

MSX・M-80の実行

- 4.1 MSX・M-80
- 4.2 MSX・M-80で使用するファイル
- 4.3 MSX・M-80の呼び出し
- 4.4 MSX・M-80コマンド
- 4.5 スイッチ
- 4.6 メッセージ
- 4.7 リスティングフォーマット
- 4.8 エラーコードとエラーメッセージ

第4章

MSX・M-80の実行

4.1 MSX・M-80

MSX・M-80 はソースファイルに作成したソースプログラムをアセンブルし、リロケートブルな（再配置可能）なオブジェクトプログラムを作成します。オブジェクトプログラムはディスクファイルに保存することができます。また、オブジェクトプログラムを保存するディスク上のファイルをオブジェクトファイルと呼び、指定がない限りファイル名拡張子を .REL にします。

オブジェクトプログラム（.REL）は実行可能ではありません。オブジェクトプログラムは MSX・L-80 で処理後、はじめて実行可能となります。

パス

MSX・M-80 はパス 1 とパス 2 の 2 段階でソースプログラムをアセンブルします。

パス 1 ではプログラム中のステートメントの評価をし、オブジェクトコード生成量を計算します。また、プログラム中のシンボルに値を割り当て、シンボルテーブルを生成します。プログラム中にマクロ呼び出しステートメントがあればマクロを展開します。

パス 2 ではパス 1 で生成したシンボルテーブルを使用し、オブジェクトコード中のシンボルや式の参照箇所にはシンボルや式を代入します。

また、マクロを再び展開し、オブジェクトファイル（.REL）にリロケートブルなオブジェクトコードを出力します。

MSX・M-80 はパス 1、パス 2 でシンボル、式、およびマクロの値をチェックします。パス 2 での値がパス 1 での値と異なる場合、フェーズエラーとなります。

4.2 MSX・M-80で使用するファイル

MSX・M-80 はアセンブル時に次のファイルを使用します。これらのファイル名は 4.3, 4.4 で述べる MSX・M-80 コマンドで指定します。

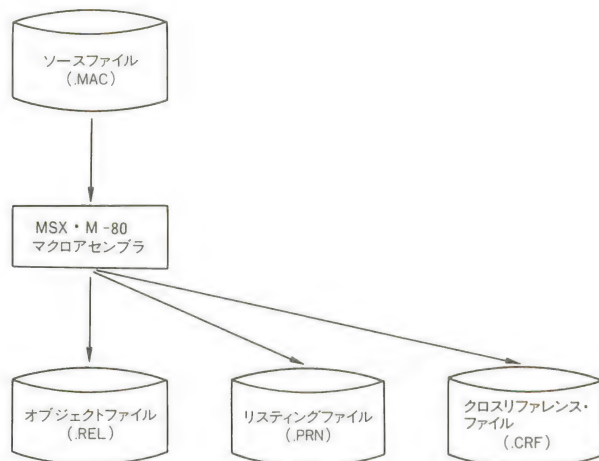


図 4-1 MSX・M-80 で使用するファイル

- (1) ソースファイル
アセンブリ言語で書かれたソースプログラムが入っているファイル。汎用テキストエディタで作成します。
- (2) オブジェクトファイル
MSX・M-80 でアセンブルした結果であるオブジェクトプログラムを入れるファイル。
- (3) リスティングファイル
ソースプログラムと生成したオブジェクトコードのリスティングファイルです。
- (4) クロスリファレンス・ファイル
MSX・M-80 では /C スイッチを指定することにより、ディスク上へクロスリファレンス・ファイルを作成することができます。クロスリファレンス・ファイルにはリスティングファイルの情報に加えて内部ラベルのクロスリファレンス情報が含まれており、CREF-80 クロスリファレンサを使ってクロスリファレンス・リストを得ることができます。

4.3 MSX・M-80の呼び出し

MSX・M-80 はコマンドで呼び出します。また、このときアセンブル中に使うファイル名や MSX・M-80 に対するスイッチを MSX・M-80 コマンドで指定します。

MSX・M-80 の呼び出し方法は次の 2 通りがあります。

- (1) 呼び出し時のコマンドに MSX・M-80 コマンドを指定する。

A>M80 <MSX・M-80 コマンド>

- この呼び出し方法はプログラムを 1 つだけアセンブルする場合に便利です。この呼び出し方法ではタイピングの手間が少なくて済みます。
 - この呼び出し方法を使ってアセンブル処理を BATCH コマンドの 1 部とすることができます。
 - プログラムのアセンブル後、制御は MSX・M-80 からオペレーティングシステムへ自動的に戻ります。
 - コマンドや MSX・M-80 コマンドは大文字で入力してください。
- (2) 呼び出し時のコマンドにパラメータを指定せず、コンパイラのプロンプトに応じて MSX・M-80 コマンドを指定する。

A>M80

* <MSX・M-80 コマンド>

- この呼び出し方法により 1 台のみのドライブ構成で MSX・M-80 を使用することができます。
- また、MSX・M-80 を再呼び出しせずに複数のプログラムを連続してアセンブルすることができます。1 つのプログラムのアセンブルが完了すると、MSX・M-80 はアスタリスク (*) プロンプトを表示して次の MSX・M-80 コマンドを待ちます。
- プロンプトが表示されている時に MSX・M-80 を終了する場合は **CONTROL**+**C** をキーインします。
- MSX・M-80 コマンドは大文字で入力しなければなりません。小文字で入力すると ?Command というエラーメッセージが表示されます。

4.4 MSX・M-80コマンド

4.4.1 MSX・M-80 コマンドの形式

MSX・M-80 コマンドには、アセンブル時に使用するファイルとスイッチを指定します。コマンドの形式は次のとおりです。

[〈オブジェクトファイル名〉] [, [〈リスティングファイル名〉]] = 〈ソースファイル名〉 / 〈スイッチ〉

〈オブジェクトファイル名〉, 〈リスティングファイル名〉, 〈ソースファイル名〉は4.2で述べたファイル名や論理デバイス名を指定します。

4.4.2 ファイル名の指定

ファイル名の指定方法は次のとおりです。

〈デバイス名〉: 〈ファイル名〉. 〈拡張子〉

例) A: SAMPLE. MAC

デバイス名

デバイス名はファイルの入出力装置の指定です。デバイス名は1～3文字で、直後にコロン(:)を付けます。デバイス名を省略するとデフォルトドライブを仮定します。従って、ファイルがデフォルトドライブ上にある場合デバイス名の指定は必要ありません。MSX・M-80で使用できるデバイス名は次のとおりです。

指定	デバイス
A:, B:, C:, ...	ディスクドライブ
PRN	プリンタ
CON	コンソールのスクリーンまたはキーボード

拡張子

拡張子は1～3文字の英数字です。拡張子の前にはピリオド(.)を置きます。拡張子のないファイル名の後にはピリオドを指定します。

例) A : SAMPLE .

拡張子は付けなくてもよく、また任意の名称にすることができますが、拡張子を省略すると標準的な拡張子を仮定しますので次のようなMSX-DOSでの標準的な拡張子を付けることをおすすめします。

拡張子	ファイルの種類
. MAC	MSX・M-80 ソースファイル
. REL	オブジェクトファイル
. COM	実行可能ファイル
. PRN	リスティングファイル
. C	C ソースファイル

MSX・M-80 スイッチはコマンドの後に任意の数だけ指定することができます。スイッチについては4.5で述べます。

例) A> M80 SAMPLE, SAMPLE = SAMPLE/R/C

ファイル名はその一部、あるいは全部を省略することができます。次にコンパイラで使用するファイルとその省略の方法を述べます。

(1) ソースファイル

ソースファイルの指定は省略することができません。しかし、ファイル名のうちデバイス名と拡張子を省略することができます。デバイス名の省略時にはデフォルトドライブを、また、拡張子の省略時には、MACを指定したものとみなします。拡張子が、MACでない場合拡張子を省略することはできません。デフォルトドライブがA:であるとき次のファイル名は同じです。

例) A : SAMPLE . MAC

A : SAMPLE

SAMPLE . MAC

SAMPLE

(2) オブジェクトファイル

オブジェクトファイルの指定はオプションです。デバイス名を省略するとデフォルトドライブを指定したとみなされます。また、拡張子を省略すると、REL を指定したとみなされます。

オブジェクトファイル名とリスティングファイル名を省略すると、拡張子が、REL であることを除いてソースファイルと同じファイル名を仮定します。オブジェクトファイルをソースファイル名と異なるファイル名にする場合、オブジェクトファイル名の指定は省略できません。

また、オブジェクトファイルとリスティングファイルを作成する場合、オブジェクトファイルの省略はできません。デフォルトドライブが A : のとき次のファイル名は同じです。

例) A : SAMPLE . REL

A : SAMPLE

SAMPLE

(3) リスティングファイル

リスティングファイルの指定はオプションです。リスティングファイル名の前にはカンマ (,) を置きます。リスティングファイル名を省略すると、/L スイッチを指定していない限りリスティングファイルは作成されません。/L スイッチについては 4.5 を参照ください。

デバイス名を省略するとデフォルトドライブを指定したものとみなし、拡張子を省略すると、PRN を指定したものとみなされます。

4.4.3 MSX・M-80 コマンドの例

MSX・M-80 コマンドは次のように省略することができます。

- オブジェクトファイルのみ作成する。

= <ソースファイル名>

<オブジェクトファイル名> = <ソースファイル名>

= <ソースファイル名> のみの場合、オブジェクトファイルがソースファイルと同じディスク上に作成されます。オブジェクトファイル名は拡張子が、REL であることを除いてソースファイル名と同じです。

例) M80 = NEIL

上の例はソースファイル NEIL . MAC を入力し、オブジェクトファイル NEIL . REL を出力します。

- オブジェクトファイルやリスティングファイルを作成しない。

, = <ソースファイル名>

MSX・M-80はソースプログラムのシンタックスチェックを行います。オブジェクトファイルやリスティングファイルを作成しないのでアセンブル時間が短く、エラーをより早く知ることができます。

例) M80 , = NEIL

4.5 スイッチ

スイッチは MSX・M-80 に対して追加機能や代替機能を指示します。スイッチは MSX・M-80 コマンドの最後に指定します。スイッチはいくつでも指定することができますが、各スイッチの前にはスラッシュ（/）を入れます。

例) M80 = NEIL/L/R

MSX・M-80 で使用できるスイッチは次のとおりです。

表 4-1 スイッチ一覧

スイッチ	説 明
/O	リスティングファイルのアドレス表示を 8 進数にする。
/R	<p>ソースファイルと同じ名前（拡張子は .REL）のオブジェクトファイルをソースファイルと同じディスク上に作成します。/R は MSX・M-80 コマンドで、オブジェクトファイル名を指定する代わりに使用できます。次の 2 つの例では NEIL.REL というオブジェクトファイルが作成されます。</p> <p>例) M80, NEIL=NEIL/R M80, =NEIL/R</p>
/L	<p>ソースファイルと同じ名前（拡張子は .PRN）のリスティングファイルをソースファイルと同じディスク上に作成します。/L は MSX・M-80 コマンドで、リスティングファイル名を指定する代わりに使用できます。次の 3 つの例では NEIL.PRN というリスティングファイルが作成されます。</p> <p>例) M80 =NEIL/L M80 , =NEIL/L M80 NEIL=NEIL/L</p>
/C	CREF-80 クロスリファレンス機能で使用する、クロスリファレンス情報ファイルをソースファイルと同じ名前（拡張子は .CRF）にして作成します。クロスリファレンス機能を使用する場合、/C スイッチを指定してアセンブルしなければなりません。詳細は第 6 章「CREF-80 クロスリファレンサ」を参照してください。
/Z	Z80 オペコードのアセンブルを指示します。/Z はプログラム中で .Z80 擬似命令を指定したのと同じ働きをします。

スイッチ	説 明
／I	8080オペコードのアセンブルを指示します。／Iはプログラム中で .8080擬似命令を指定したのと同じ働きをします。／Zや／Iスイッチを指定していない場合、／Iを指定したものとみなされます。
／P	／Pを指定するごとに、アセンブル時に使用するスタック領域を255バイト追加割り当てします。／Pはアセンブル時にスタック・オーバーフローエラーが起った場合に使用し、それ以外の場合には必要ありません。
／M	D S（領域定義）擬似命令によって定義するデータエリアをゼロに初期設定します。／Mスイッチを指定しない場合、このデータエリアはゼロに初期設定されず、プログラム開始時のデータエリアの内容は保証されません。
／X	偽条件のリスティングを抑止します。／Xの指定がない場合、偽条件ブロックをリスティングします。／Xは.TFCOND擬似命令と関連して使用される場合があります。詳細は3.6.6リスティング擬似命令を参照ください。

4.6 メッセージ

アセンブル終了後に次のメッセージが出力されます。

$\left\{ \begin{array}{l} \text{xx} \\ \text{No} \end{array} \right\}$ Fatal errors [yy warning]

xx は重大な文法エラーの件数です。また yy は警告エラーです。メッセージはアセンブル終了時にコンソールおよびリスティングファイルに表示されます。プログラムにエラーがなければ "No Fatal Error" が出力されます。

4.7 リスティングフォーマット

MSX・M-80 のリストは次の 2 つから構成されています。

(1) アセンブリリスト

ソースプログラムと生成したオブジェクトコードがプリントしてあります。また、プログラム中にシンタックスエラーがあれば、エラーコードがエラーの発生した行に表示されます。

(2) シンボルテーブルリスト

プログラム中のすべてのマクロ名とシンボルがそれぞれアルファベット順にリストされます。

MSX・M-80 のリスティングは、1 ページに最大行数をプリントした後、または PAGE 擬似命令ごとに改頁します。

1 ページの最大行数は PAGE 擬似命令で指定していればその行数、また、指定がなければ 50 行が指定されたものとみなされます。

以下、リスティングフォーマットについて解説します。サンプルリストを参考にしてください。

① 見出し

各ページの先頭 2 行には見出しがプリントされます。見出しの形式は次のとおりです。

[タイトル] MSX・M-80 z . zz PAGE x

[サブタイトル]

② タイトル

TITLE 擬似命令で指定したタイトルがプリントされます。TITLE 擬似命令を指定しない場合は空白となります。

③ サブタイトル

SUBTITLE 擬似命令で指定したサブタイトルがプリントされます。SUBTITLE 擬似命令を指定しない場合は空白となります。

④ バージョン番号

プログラムのバージョン番号をプリントします。

MSX・M-80 z . zz (z . zz はバージョン番号)

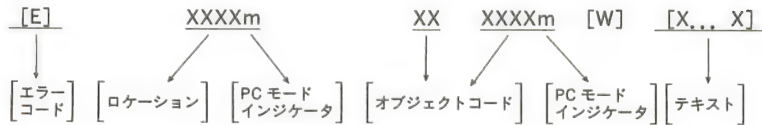
⑤ ページ番号

PAGE x (x はページ番号)

ただしシンボルテーブルリストでは PAGE S と表示されます。

⑥ アセンブルリスト

アセンブルリストにはプログラムのテキストが次の形式でプリントされます。



⑧ エラーコード

ソースステートメントにエラーがある場合だけ1文字のエラーコードがプリントされます。エラーコードの詳細は4.8を参照ください。

⑨ ロケーション

セグメント(PCモードに対応している)の先頭からのロケーションをプリントします。ロケーションはMSX・M-80コマンドで/Oスイッチを指定している場合は8進数で表示され、/Oを指定していない場合は16進数で表示されます。

⑩ PCモードインジケータ

PCモードによって次のようなインジケータがプリントされます。

' コード相対モード
" データ相対モード
! COMMON 相対モード
空白 絶対モード
* 外部参照

⑪ オブジェクトコード

生成したオブジェクトコードが16進数でプリントされます。

X X	X X X X
↓	↓ ↓ ↓
1バイト目	3バイト目 2バイト目

××××では上位バイトと下位バイトとが実際に生成されるオブジェクトコードとは逆に表示されます。

⑫ [W]

INCLUDE 擬似命令で他のファイルから呼び出して展開したステートメントにはCが、またMACRO, REPT, IRP, IRPC 擬似命令で展開したステートメントには+が表示されます。

⑬ テキスト

ソースプログラムのテキストがプリントされます。

⑭ シンボル

シンボルは次のとおりプリントされます。

<u>X</u>	<u>X</u>	<u>X</u>	<u>X</u>	<u>X</u>		<u>X</u>	<u>X</u>	<u>...</u>	<u>X</u>
				↓		↓			↓
				値		シンボルの種類			シンボル

⑮ シンボルの値

MSX・M-80 がシンボルに対して割り当てた値をプリントします。

⑯ シンボルの種類

シンボルの種類によって次のような文字が表示されます。

I PUBLIC シンボル

U 未定義シンボル

C COMMON ブロック名

COMMON ブロックのサイズが 16 進または 8 進数のバイト数で表示されます。

* 外部シンボル

空白 絶対値

' コード相対値

" データ相対値

! COMMON 相対値

4.8 エラーコードとエラーメッセージ

プログラム中にエラーがある場合、リスティングファイルとコンソールにエラーコードまたはエラーメッセージが表示されます。

エラーコードはリスティングファイルの該当ステートメントの行のカラム 1 に表示されます。また、エラーメッセージはリスティングファイルの終りにプリントされます。

エラーコードの付いた行やエラーメッセージは、リスティングファイルの出力とは別にコンソールに出力されます。

表 4-2 エラーコード一覧

エラーコード	説 明
A	Argument error 擬似命令に対するアーギュメントの形式が正しくないか、指定値が範囲外である。
C	Conditional nesting error IFのないELSE、IFのないENDIF、IFに対して2つのELSEがある、CONDのないENDCなどの場合です。
D	Double Defind symbol 参照するシンボルが2箇所以上で定義されている。
E	External error エクスターナルシンボルが誤って使用されている。 例) F00 SET NAME##; LXI B, 2-NAME
M	Multiply Defined symbol 定義しようとしたシンボルはすでに定義済みです。
N	Number error 数値の表現が間違っています。 例) 8 Q
O	Bad opcode or objectionable syntax • ENDMやLOCAL がマクロ定義ブロック外にある。 • SET, EQU, MACROにシンボル名やマクロ名を指定していない。 • オペコードが間違っているか、式の書式が間違っています。 例) かっこや引用符が対になっていない、演算子が連続している。
P	Phase error ラベル、またはEQUのシンボル名の値がパス1とパス2とで異なる。

エラーコード	説 明
Q	<p>Questionable 行が正しく終了していない場合などです。これは警告エラーです。</p> <p>例) MOV AX, BX,</p>
R	<p>Relocation 式内のリロケーションを間違えて使用した場合など。</p> <p>例) <アブソリュート>-<リラティブ></p> <p>データセグメント、コードセグメント、COMMONセグメントの各領域はリロケートابلです。</p>
U	<p>Undefined symbol 式中で使用しているシンボルは定義されていません。この場合、パス1でVエラーが表示され、パス2でUエラーが表示されます。Vエラーの説明を参照ください。</p>
V	<p>Value error パス1で決まる値を持たなければならない擬似命令（たとえば .RADIX, .PAGE, DS, IF, IFE など）が定義されていない値を持っている。 このステートメント以降でシンボルが定義されていればUエラーは表示されません。</p>

表 4-3 エラーメッセージ一覧

エラーメッセージ	説 明
% No END statement	END文がない。またはEND文が偽条件IRP/IRPC/REPTブロックやマクロ定義ブロックの中で定義されている。
Unterminated conditional	条件ブロックが終了していない。
Unterminated REPT/IRP/IRPC/MACRO	反復擬似命令のブロックが終了していない。
Symbol table full	シンボルテーブルを作成中にメモリが不足した。一般的にマクロ定義ブロックの数が多い場合に発生します。マクロ定義ブロックは内部のコメントを含めてシンボルテーブルにストアされるので、コメントの前のセミコロン（;）をダブルセミコロン（;;）に変えてみてください。
[xx] [No] Fatal errors [.yy warnings]	xxは重大なエラーの数、また、yyは警告エラーの数です。このメッセージはアセンブル終了時にコンソールとリストニングファイルへ出力されます。

第5章

MSX・L-80リンクローダ

- 5.1 MSX・L-80
- 5.2 MSX・L-80で使用するファイル
- 5.3 MSX・L-80の呼び出し
- 5.4 MSX・L-80コマンド
- 5.5 スイッチ
- 5.6 メッセージ
- 5.7 エラーメッセージ

第5章

MSX・L-80 リンクローダ

5.1 MSX・L-80

MSX・M-80 や他のコンパイラで作成したオブジェクトファイル(.REL ファイル)はまだ実行可能ではありません。MSX・L-80 リンクローダは、.REL ファイルに入っているオブジェクトモジュールをリンクロードし、実行可能なプログラムに変換してそのまま実行したり実行可能ファイル(.COM ファイル)としてディスクにセーブしたりします。

リンクローダは次の機能を持っています。

- 相対アドレスを絶対アドレスに変換する。
アセンブラやコンパイラで作成したオブジェクトモジュールにはラベルや変数名がモジュール内での相対アドレスの形で含まれています。MSX・L-80 はオブジェクトモジュールをメモリ上にロードし、相対アドレスへロードアドレスを加えて絶対アドレスに変換します。
- モジュール間のグローバルリファレンス（外部参照）を解決する。
あるモジュールから別のモジュールを呼び出したり、別のモジュールのラベルや変数名を参照することをグローバルリファレンス（外部参照）といいます。
リンクロードしようとするモジュールが別にアセンブルまたはコンパイルしておいたモジュールを呼び出したり参照したりする場合(モジュール中に CALL 文や EXTERNAL シンボルがある場合)、呼び出し先や参照先の絶対アドレスが呼び出し元や参照元へ代入されます。
- 高級言語で作成したプログラムにランタイムライブラリをリンクする。
BASIC コンパイラ、COBOL-80、FORTRAN-80 などのコンパイラで作成したオブジェクトモジュールにランタイムライブラリをリンクします。高級言語用のランタイムライブラリやランタイムライブラリのリンク方法については、各コンパイラのユーザーズ・マニュアルを参照してください。
- 実行可能プログラムのセーブ
リンクロード後の実行可能プログラムをディスク上の実行可能ファイル(.COM ファイル)へセーブすることができます。セーブした実行可能プログラムはオペレーティングシステムのコマンドレベルでファイル名を指定するだけで呼び出すことができます(このとき、拡張子.COM を入力する必要はありません)。

例) B: SAMPLE

5.2 MSX・L-80で使用するファイル

リンクロード時に使用するファイルは次のとおりです。

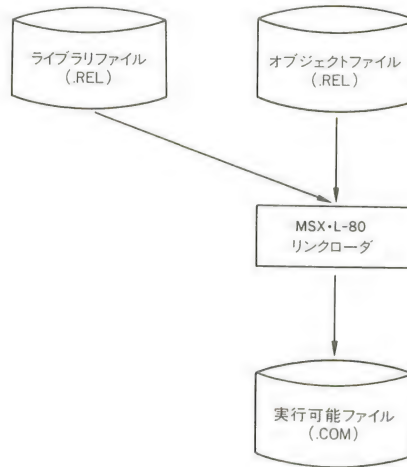


図 5-1 MSX・L-80 で使用するファイル

(1) オブジェクトファイル

コンパイル，あるいはアセンブル済みのオブジェクトモジュール（オブジェクトプログラム）が入っているファイル。省略時の拡張子は，REL です。

リンクロードでは複数個のオブジェクトファイルの入力が可能です。オブジェクトファイルを指定しないと“NOTHING LOADED” のメッセージが表示されます。

(2) 実行可能ファイル

リンクロード後の実行可能プログラムが入るファイル。省略時の拡張子は，COM。実行可能ファイルの指定はオプションです。

(3) ライブラリファイル

オブジェクトファイルを LIB-80 ライブラリマネージャを使って集め，ライブラリ形式にしたもの。ライブラリファイルの指定があり，リンクロード時に未解決のグローバルリファレンス（外部参照）があれば，ライブラリファイルが検索され必要なオブジェクトモジュールがメモリ上にロードされます。ライブラリファイルの指定はオプションです。また，FORTRAN-80，COBOL-80，BASIC コンパイラのランタイムモジュール・ライブラリについては自動的に省略時のファイルが検索されます。詳細については各コンパイラのユーザーズ・マニュアルを参照してください。

5.3 MSX・L-80の呼び出し

MSX・L-80 リンクローダ はコマンドで呼び出します。また、リンクロード時に使うファイル名やリンクローダに対するスイッチは MSX・L-80 コマンドで指定します。MSX・L-80 コマンドの指定の方法は次の2通りがあります。

- (1) リンクローダを呼び出すコマンドに MSX・L-80 コマンド列を指定する。

A > L80 <MSX・L-80 コマンド列>

例) A > L80 MYPROG, SUB1, MYPROG/N/E

- (2) リンクローダを呼び出すコマンドには MSX・L-80 コマンド列を指定せず、リンクローダのプロンプト(*)に応じて指定する。

A > L80

* <MSX・L-80 コマンド列>

* <MSX・L-80 コマンド列>

・
・
・

MSX・L-80 コマンドをリンクローダのプロンプトに応じて指定する場合、コマンドをいくつかに分けて指定することができます。リンクロードの終了は/E、/G スイッチを指定した時です。プロンプトはリンクロードを終了するまで表示されます。次の例は(1)と同じ結果になります。

例) A > L80

* MYPROG

* SUB1

* MYPROG/N

* /E

ディスクドライブが1台のときにリンクしようとするオブジェクトファイルが複数のディスクに入っている場合や、実行可能ファイルをオブジェクトファイルとは別のディスクに作成する場合、異なるディスクへの入出力のためにディスクドライブのディスクを交換しなければなりません。

5.4 MSX・L-80コマンド

5.4.1 MSX・L-80 コマンドの形式

MSX・L-80 コマンドでは、リンクロード時に使用するファイルとリンクロードに対するスイッチを指定します。MSX・L-80 コマンドの形式は次のとおりです。

〈オブジェクトファイル名〉 [, 〈オブジェクトファイル名〉]

スイッチの指定方法については 5.5 で説明します。

5.4.2 ファイル名の指定

ファイル名の指定方法は次のとおりです。

〈デバイス名〉 : 〈ファイル名〉. 〈拡張子〉

例) A : SAMPLE . REL

- デバイス名はファイルの入出力装置の指定です。デバイス名の直後にはコロン (:) を付けます。デバイス名を省略するとデフォルトドライブが仮定されます。
- 拡張子は 3 文字以内の英数字です。拡張子の前にはピリオドを置きます。

例) A : SAMPLE .

拡張子は任意の名称にすることも、また付けないこともできますが、省略すると次のような MSX-DOS の標準的な拡張子が仮定されますので、これに合わせた拡張子を付けることをおすすめします。

拡張子	ファイルの種類
. REL	オブジェクトファイル
. COM	実行可能ファイル

5.4.3 ローディングの順序

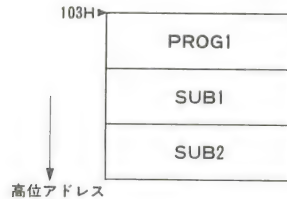
ローディングは次の規則に従って行われます。

- MSX・L-80 コマンドに指定した順序でオブジェクトファイルのロード、またはスイッチの機能が実行されます。ただし /N, /X, /Y スイッチの処理は /E または /G スイッチ実行時に行われます。スイッチの詳細については 5.5 を参照ください。
- オブジェクトモジュールのロード開始アドレスは /P, /D スイッチの指定をしない限り 103H からです。オブジェクトファイルを MSX・L-80 コマンドに指定するたびにオブジェクトモジュールが順次隣接してロードされます。しかし、モジュールの実行順序でファイル名を指定する必要はありません。

- /E または /G スイッチを指定すると、それまでにロードしたモジュール間のグローバルリファレンス(外部参照)の解決を行い、実行可能プログラムの実行開始アドレスへのジャンプ命令を 100 H~102 H へ設定します。実行開始アドレスはオブジェクトモジュールの配置とは関係ありません。

例) L80 PROG1, SUB1, SUB2, PROG1/N/E

実行可能プログラム PROG1 のローディング結果



5.4.4 MSX・L-80 コマンドの例

- オブジェクトファイルのリンクロード後実行可能プログラムを実行する。

例) * NEIL/G

NEIL.REL をリンクロードし実行します。実行可能ファイルは作成されません。

- オブジェクトファイルのリンクロード後、実行可能プログラムを実行可能ファイルへセーブし実行する。

例) * NEIL, NEIL/N/G

NEIL.REL をリンクロード後、NEIL.COM へセーブし実行する。

- 複数のオブジェクトファイルをリンクロード後、実行可能プログラムを実行可能ファイルへセーブし実行する。

例) * NEIL/N, BASPROG, ASMSUB1, ASMSUB2/G

ASMSUB1.REL, ASMSUB2.REL, BASPROG.REL をリンクロード後、NEIL.COM へセーブし実行します。

5.5 スイッチ

MSX・L-80 スイッチはリンクローダの動作を指示します。コマンド列には任意の数のスイッチを置くことができますが、スイッチの前にはスラッシュ（/）を入れなければなりません。

例) * NEIL, NEIL/N/G

以下の表は、各スイッチを機能別にまとめたものです。各スイッチの詳細については表の次に解説します。

表 5-1 スイッチ一覧

機 能	スイッチ	説 明
プログラムの実行	/ G / G : <名前>	ロードしたオブジェクトモジュールをリンク・実行し、次にオペレーティングシステムへ抜け出す。 <名前>を指定すると、プログラム開始アドレスが<名前>で指定したパブリックシンボルのアドレスに設定されます。
MSX・L-80 の終了	/ E / E : <名前>	ロードしたオブジェクトモジュールをリンクした後、オペレーティングシステムへ抜け出す。 <名前>を指定すると、プログラム開始アドレスが<名前>で指定したパブリックシンボルのアドレスに設定されます。
プログラムのセーブ	/ N / N : P	実行可能プログラムを/Nの直前のファイル名でディスクにセーブします。 : Pを指定すると、プログラム領域（CSEGセグメント）のみをセーブします。: Pの指定は/Xを指定したときのみ有効です。
ライブラリサーチ	/ S	/ Sの直前のファイル名のオブジェクトライブラリを検索します。
グローバル リスティング	/ U	未定義の外部参照名をリスティングします。
	/ M	外部参照マップをリスティングします。
基数の設定	/ O	基数を8進にします。
	/ H	基数を16進（デフォルト値）にします。
開始アドレスの 設定	/ P	オブジェクトモジュールのロード開始アドレスを設定します。/ Dスイッチと共に使用する場合、/ Pはプログラム領域（CSEG）のロード開始アドレスを設定します。
	/ D	データ領域（COMMON, DSEG）のロード開始アドレスを設定します。
	/ R	MSX・L-80 を初期状態にリセットします。

機 能	スイッチ	説 明
特殊コード	/X	Intel ASCII16進形式で実行可能ファイルを作成します。/Xスイッチは/ Nスイッチと共に指定します。実行可能ファイルの拡張子はHEXです。
	/Y	シンボルテーブルを出力します。

スイッチの指定位置はスイッチの種類によってつぎのように異なります。

- (1) コマンド列の先頭、最後、オブジェクトファイル名の後の任意の場所に置くことができる。また、独立して1つのコマンドとして指定することもできる。

(/E,/G,/R,/U,/M,/O,/H)

例) * /G

* /M

- (2) オブジェクトファイル名の後に付加し、そのファイルについて作用する。

(/N,/S,/X,/Y)

例) * MYLIB/S, MYPROG

* MYPROG, MYPROG/N

- (3) そのスイッチを作用させたいオブジェクトファイルの前に指定する。

(/P,/D)

例) * /P : 200, F00

/G

/G スwitchはコマンド行に指定したオブジェクトファイルをロード、リンク後、プログラムを実行し、その後に制御をオペレーティングシステムのコマンドレベルへ戻すことを指示します。

- 次の例は NEIL . REL をロード、リンクし、NEIL . COM というファイル名で実行可能のプログラムをディスク上にセーブします。また、リンク後のプログラムが実行され、次にオペレーティングシステムへ抜け出ます。プログラムの実行直前に5.6で説明するメッセージと“BEGIN EXECUTION”メッセージが出力されます。

例) L80 NEIL, NEIL/N/G

- 実行可能プログラムをセーブする必要がない場合は次のとおりになります。

例) L80 NEIL/G

上の例の場合 /N スイッチを指定していないので、実行可能ファイルは作成されません。このプログラムをもう一度実行するためには MSX・L-80 をもう一度実行しなくてはなりません。

/G: <名前>

/G: <名前> スイッチは /G スイッチの機能に加えて、プログラムの実行開始アドレスを設定します。

- <名前> はロードしたモジュールの 1 つにあらかじめ定義されているパブリックシンボル (PUBLIC ステートメントで宣言したラベル) です。
- MSX・L-80 はラベルとして <名前> を持つ行をプログラムの実行開始位置とし、<名前> のアドレスへのジャンプ命令が 100H~102H にセットされます。
- MSX・L-80 は自動的につぎのようなプログラム実行開始アドレスを設定しますので、通常は:<名前> の指定は必要ありません。
 - a. 高級言語 (FORTRAN, BASIC, COBOL) のモジュールをリンクロードする場合は高級言語のモジュールをメインプログラムとみなし、このモジュールの先頭の命令をプログラム開始アドレスとします。
 - b. アセンブリ言語のモジュールをリンクロードする場合は、END <式> ステートメントを持つモジュールをメインプログラムとみなし、<式> をプログラム開始アドレスとします。
- アセンブリ言語のモジュールをリンクロードするとき、次のような場合に:<名前> の指定をする必要があります。
 - a. END <式> ステートメントを含むアセンブリ言語モジュールが 2 つ以上ある場合。
 - b. END <式> ステートメントを含むアセンブリ言語モジュールが 1 つもない場合。
- 高級言語プログラムの実行に先立ってアセンブリ言語のモジュールの実行をしたい場合、高級言語モジュールの開始点で CALL 文などを使用してアセンブリ言語モジュールを呼び出してください。

/E

/E スイッチはリンクロードを終了後、オペレーティングシステムへ抜け出すことを指示します。

- リンクロードを終了後プログラムの実行をしない場合 /E スイッチを使用します。MSX・L-80 の終了は /E スイッチの他に /G スイッチのみが使用できます。リンクロード終了直後に 5.6 で説明するメッセージが出力されます。
- /G スイッチの項を参照してください。

／E：〈名前〉

／E：〈名前〉スイッチは／Eスイッチの機能に加えて、プログラムの実行開始アドレスを設定します。

- 〈名前〉はロードしたモジュールの1つにあらかじめ定義されているパブリックシンボル（PUBLIC ステートメントで宣言したラベル）です。MSX・L-80はラベルとして〈名前〉を持つ行をプログラムの実行開始位置とし、〈名前〉のアドレスへのジャンプ命令が100 H～102 Hにセットされます。
- ：〈名前〉の指定は／G：〈名前〉スイッチの項で説明しているような場合のアセンブリ言語モジュールをリンクロードするときに必要になります。

／N

／Nスイッチは、リンクロード後の実行可能プログラムを／Nの直前に指定したファイル名でディスク上にセーブします。

- 〈ファイル名〉は出力となる実行可能ファイル名です。〈ファイル名〉の拡張子を省略すると、COMが仮定されます。このファイル（実行可能ファイル）をCOMファイルと呼びます。COMファイル名は必ず／Nスイッチの直前に付与してください。
- ／Nスイッチの動作は／Eまたは／Gスイッチの機能を実行するまで据え置かれます。／Nスイッチは、／Nスイッチの指定以降に／Eまたは／Gの指定がない限り効果はありません。
- ／Nスイッチを指定する場合、MSX・L-80コマンドには1つ以上のリンクロードすべきオブジェクトファイル名と、ディスクへセーブするリンクロード後の実行可能ファイル名を指定しなければなりません。

例) L80 NEIL, NEIL／N／G

上の例は最低限必要な項目を指定したコマンドです。最初のファイル名NEILはリンクロードしようとするオブジェクトファイル名です。2番目の／Nを指定したファイル名NEILはディスクにセーブするリンクロードの結果の実行可能ファイル名（COMファイル名）です。

- また、コマンドに指定するファイル名の順序は任意です。

例) L80 NEIL／N, BASPROG, ASMSUB1, ASMSUB2／G

上の例はオブジェクトファイルのBASPROG, ASMSUB1, およびASMSUB2をロード、リンクし、NEILというファイル名で実行可能ファイルをセーブします。

- ／Nスイッチを指定して実行可能ファイルをセーブしないと、リンクロード終了後にプログラムを実行する場合に再度リンクロードしなければなりません。

/N:P

リンクロード後の実行可能プログラムのうちプログラム領域 (CSEG エリア) のみを /N スイッチの直前のファイル名でディスク上にセーブします。:P の指定は、/X を指定して Intel ASCII 16 進形式のファイルを作成するときのみ有効です。

- <ファイル名>/N は実行可能ファイルのプログラム領域とデータ領域の両方をディスクにセーブしますが、実行可能ファイルのサイズを小さくする場合などでは /N:P の形式にすることによりプログラム領域部分だけをディスク上にセーブすることができます。

/P: <アドレス>

/P スイッチは、/P スイッチ以降に指定するオブジェクトモジュールのロード開始アドレスを指定します。また、/D スイッチを共に指定しているとき、オブジェクトモジュールのうちプログラム領域 (CSEG エリア) のロード開始アドレスを指定します。

- <アドレス> はカレント基数、つまり /O スイッチの指定があれば 8 進で指定し、/H スイッチの指定があれば 16 進で指定します。また /O も /H も指定していない場合 16 進で指定します。
- <アドレス> の後はカンマ (,) で区切ります。
- /P スイッチを指定しない場合、デフォルトのアドレス 103 H からオブジェクトモジュールを配置します。
- <アドレス> はロード開始アドレスです。

例) L80 /P: 103, NEIL, NEIL/N/E

- /P スイッチを指定し、/D スイッチを指定していない場合、オブジェクトモジュールのうちデータ領域 (DSEG エリアと COMMON エリア) が /P で指定した <アドレス> へロードされ、その後へ隣接してプログラム領域がロードされます。
- /P スイッチは /P スイッチの後に指定するファイルに対して有効ですが、/P スイッチの前に指定したファイルには無効です。/P スイッチは指定したアドレスでロードを開始したいファイルの前に置きます。
- /P スイッチと /D スイッチを指定している場合、オブジェクトモジュールのうちプログラム領域が /P スイッチに指定した <アドレス> へロードされます。また、データ領域 (DSEG エリアと COMMON エリア) とが /D スイッチに指定した <アドレス> からロードされます。プログラムが CSEG に加えて DSEG, あるいは COMMON から構成されているとき、/P と /D スイッチを一緒に使用することができます。
- ASEG エリアのロードアドレスは /P スイッチの影響を受けません。
- MSX・L-80 ではセグメントが自動的に順次隣接して配置されますが、同一リンクロードセッション中に 2 つ以上の /P スイッチを指定して複数のプログラム領域を隣接しないアドレスに配置し、モジュール間に空スペースを作ることができます。ただし空スペースは初期化されません。

- モジュールの配置については次の制約があります。
 - a. ロードしようとする複数のプログラム領域が重複しないようにアドレスの指定をしてください。
重複すると警告エラーメッセージが表示されます。
 - b. プログラムコードエリアがデータエリア (DSEG エリア) またはコモンエリア (COMMON エリア) によって分割されてはいけません。たとえば 200 H から CSEG を配置し、300 H から DSEG を配置し、400 H から CSEG を配置したりしてはいけません。このような場合重大エラーメッセージが表示されます。

データ領域のサイズはアセンブルのリスティングの最後を見て DSEG エリアと COMMON エリアのサイズを加えて求めます。

- リンクロード時にメモリが不足してリンクロードできないプログラムを /P、/D スイッチを使うことによりリンクロードできることがあります。

MSX・L-80 はリンクロード時に相対リファレンスごとに 5 バイトで構成するテーブルを作成しますが、/D および /P を指定するとこのテーブルを作成しないので、リンクロード時の必要メモリ量を小さくすることができます。この時の /P、/D スイッチに指定するアドレスは次のとおりです。

/P スイッチで指定するアドレス=データ領域のサイズ+ 103 H

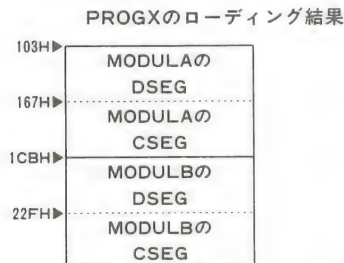
/D スイッチで指定するアドレス= 103 H

例)

リンクロード前のオブジェクトモジュール

MODULA	MODULB
DSEG (100バイト)	DSEG (100バイト)
CSEG (100バイト)	CSEG (100バイト)

- L80 MODULA, MODULB, PROGX/N/E



- L80 MODULA, /P : 400, MODULB, PROGX /N /E

PROGXのローディング結果

103H▶	MODULAの DSEG
167H▶	MODULAの CSEG
1CBH▶	空きエリア
400H▶	MODULBの DSEG
464H▶	MODULBの CSEG

- L80 /P : 150, /D : 400, MODULA, MODULB, PROGX /N /E

PROGXのローディング結果

103H▶	空きエリア
150H▶	MODULAの CSEG
1B4H▶	MODULBの CSEG
218H▶	空きエリア
400H▶	MODULAの DSEG
464H▶	MODULBの DSEG
4C8H▶	

/D : <アドレス>

/Dスイッチは/Dスイッチの後に指定するオブジェクトファイルのうちデータ領域（DSEG エリアとCOMMON エリア）を分離し、指定したアドレスへロードするように指示します。

- <アドレス> はカレント基数，つまり/O スイッチの指定があれば8進で指定し，また/H スイッチの指定があれば16進で指定します。/O も/H も指定していない場合には16進で指定します。
- <アドレス> の後はカンマ（,）で区切ります。

例) L80 /D : 103, NEIL, NEIL /N /E

- /D スイッチの指定がない場合，/P スイッチで指定したアドレスからデータ領域のロードを開始します。また，/P の指定もない場合，103 H が省略時のアドレスとなります。

- /D スイッチは/D スイッチの後に指定するファイルに対して有効です。/D スイッチの前に指定したファイルには無効です。データ領域を特定のアドレスへロードしたい場合はそのファイルの直前に/D スイッチを指定します。
- /D スイッチは/P スイッチと関連していますので、/P スイッチの項を参照してください。

/S

/S スイッチは/S スイッチの直前に指定したファイルを検索し、ファイルに含まれているモジュールを参照してグローバルリファレンスの解決をはかります。

- /S スイッチの直前のファイルは LIB-80 ライブラリマネージャを使用して作成したライブラリファイルです。
- /S スイッチの直後はカンマで区切ります。

例) L80 NEIL/N, MYLIB/S, NEIL/G

/R

/R スイッチは MSX・L-80 を初期状態にリセットします。

- MSX・L-80 は最初にコマンド行をスキャンし、/R スイッチがあればロードされているすべてのファイルを無視して MSX・L-80 を呼び出した直後の状態にリセットします。このときアスタリスク (*) プロンプトが表示され、コマンドの入力待ちとなります。

/U

/U スイッチは未定義状態のグローバルシンボルをコンソールにリストします。また、プログラム領域とデータ領域とのそれぞれの先頭アドレスと、最終アドレスおよび領域サイズを表示します。

- /U は/G または/E スイッチのどちらも含まれないコマンド行でのみ有効です。
- 未解決状態のグローバルシンボルが多い場合、スクリーンがスクロールアップしてリストが見えなくなることがあります。スクロールアップを止めたい場合は **CONTROL** + **S** を押します。また、これを解除する場合は **CONTROL** + **Q** を押します。

/M

/M スイッチはすべてのグローバルシンボルをコンソールにリストします。また、プログラム領域とデータ領域とのそれぞれの先頭アドレスと最終アドレスおよび領域サイズを表示します。

- リストはプリンタに出力することはできません。

- グローバルシンボルが定義済みであればグローバルシンボルの後にその値が表示され、また、未定義であればアスタリスク（*）が表示されます。
- プログラム領域およびデータ領域の先頭・最終アドレスとサイズの値は／P と／D スイッチを指定していないかぎり、1つのランプエリアとしてリストされます。
／P と／D スイッチを指定した場合、プログラム領域とデータ領域とがそれぞれ別々にリストされます。

／O

／O スイッチはカレント基数を 8 進にセットします。

- 基数のデフォルト値は 16 進数です。／O を省略すると 16 進が仮定されます。

／H

／H スイッチはカレント基数を 16 進にセットします。

- 基数のデフォルト値は 16 進ですので、前に／O スイッチを指定していて 16 進へ変更する場合以外には使用する必要はありません。

／X

／X スイッチは実行可能ファイルをインテル ASCII HEX フォーマットにし、／X の直前のファイル名でセーブします。

- ／X スイッチは／N スイッチを指定しているファイルに付与します。

例) L80 NEIL, NEIL／X／N／E

- ／X スイッチを指定している実行可能ファイルの省略時の拡張子は、HEX です。
- ／X の主な用途は PROM に焼き込むプログラムを用意する場合です。
- HEX フォーマットの実行可能ファイルはあるマシンから他のマシンへのプログラムの移植を容易にする目的で作成されました。HEX フォーマットの形式はオブジェクトコードよりもコードのチェックが多くなります。また、HEX ファイルは高度なラインエディタで編集することができます。

／Y

シンボルテーブルを作成し／Y の直前のファイル名でセーブします。

- ／Y スイッチは／N スイッチを指定しているファイルに付与します。また、／E スイッチの指定を必要とします。

例) L80 NEIL, NEIL／Y／N／E

- ／Y スイッチを指定している実行可能ファイルは、.COM ファイルと拡張子 .SYM を持つファイルとの2通りが作成されます。ただし、／X スイッチを同時に指定していると、.SYM ファイルと .HEX ファイルとが作成されます。

例) L80 NEIL, NEIL／Y／X／N／E

5.6 メッセージ

リンクロードを終了すると次のメッセージがコンソールに表示されます。

[mmmm nnnn yy]

mmmm ; 実行可能プログラムの先頭アドレス

nnnn ; 実行可能プログラムの最終アドレス + 1

yy ; 1 ページ / 256 バイトで表す実行可能プログラムのサイズ (ページ数)

5.7 エラーメッセージ

LINK-80 では次のようなエラーメッセージが用意されています。

表 5-2 エラーメッセージ一覧

エラーメッセージ	説 明
? No start Address	/G スイッチが指定されたが、メインプログラムがロードされていない。
? Loading Error	入力として与えられた最後のファイルが MSX・L-80 に対するオブジェクトファイルとして正しい形式ではなかった。
? Out of Memory	プログラムをロードするための十分なメモリがない。
? Command Error	MSX・L-80 のコマンドではない。
? <ファイル名> Not Found	コマンドの中で指定された<ファイル名>で示されるファイルが、どこにも存在しない。
% 2nd COMMON Larger /XXXXXX/	最初に定義したCOMMONブロック/XXXXXX/が最大のものではなかった。モジュールのローディング順序を変えるか、またはCOMMONブロックの定義を変える必要があります。
% Mult. Def. Global YYYYYY	パブリック (外部) シンボルYYYYYYが重複して定義されていることがローディング中に判明した。
% Overlaying. { Program } Area { ,start =XXXX } { ,public =<シンボル>(XXXX) } { ,External=<シンボル>(XXXX) }	/Dまたは/Pによって既にロードされているデータを壊す恐れがあります。/Dや/Pスイッチで指定したアドレスがMSX・L-80の領域を指していたり、既にロード済みの領域を指しているときに発生します。

エラーメッセージ	説 明
? Intersecting { Program } Area { Data }	プログラムとデータ領域とが重なっており、そのためアドレスまたはグローバルシンボルのチェインエントリも重なっている (だぶっている)。最終的な値がこの重なった領域にあるため、現在の値に変換することができない。
? Start symbol - 〈名前〉 -Undefined	／Eまたは／Gが指定されたが、その記号が定義されていない。
Origin { Above } Loader { Below } Memory, Move, Anyway (Y or N) ?	／Eまたは／Gが指定されたが、データ、プログラムのいずれかの領域の先頭アドレスがロードメモリー (モジュールをロードするために利用できるメモリ) の外にある。Y 〈キャリッジリターン〉 と答えた場合 MSX・L-80 はプログラムまたはデータの領域を移動させ処理を続行する。 その他の答の場合は MSX・L-80 は終了する。いずれの場合においても／Nが指定してあれば、ロードイメージは常に保存されることになる。
? Can't Save Object File	ファイルをセーブしようとしたときにディスクエラーが発生した。 ディスクがいっぱいであるか、書き込み禁止の場合に発生します。
? Nothing Loaded	オブジェクトファイルがロードされていないのに、〈ファイル名〉／S又は／E又は／Gが与えられた。 即ち、オブジェクトファイルがロードされていない時に、ライブラリの検索、リンクの処理終了、またはプログラムの実行が要求された。

第 6 章

CREF-80クロスリファレンサ

- 6.1 CREF-80
- 6.2 CREF-80で使用するファイル
- 6.3 クロスリファレンス・ファイルの作成
- 6.4 CREF-80の呼び出し
- 6.5 CREF-80コマンド
- 6.6 リスティング制御擬似命令
- 6.7 リスティングフォーマット

第6章

CREF-80 クロスリファレンサ

6.1 CREF-80

CREF-80 クロスリファレンサは MSX・M-80 で出力されたクロスリファレンス・ファイルを入力し、プログラム内の内部ラベルとその定義位置をクロスリファレンス・リストへ出力します。このクロスリファレンス・リストはデバッグに役立ちます。

6.2 CREF-80で使用するファイル

クロスリファレンサの実行時に使用するファイルは次のとおりです。

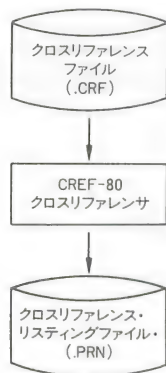


図 6-1 CREF-80で使用するファイル

(1) クロスリファレンス・ファイル

MSX・M-80 でプログラムをアセンブルした時に /C スイッチを指定して出力したリスティングファイル。クロスリファレンス・ファイルには MSX・M-80 でセットされた制御文字が組み込まれています。クロスリファレンス・ファイルの省略時の拡張子は .CRF です。

(2) クロスリファレンス・リスティングファイル

クロスリファレンス・リスティングは、MSX・M-80 のリスティングファイルに加えて次の追加情報が付加されています。

- a. ソースステートメントの各文にはクロスリファレンス番号が付与されている。
- b. リストの最後にシンボルがアルファベット順に並べてプリントされます。さらにシンボルの後にはそのシンボルを参照している行と、シンボルの定義をしている行の行番号がプリントされます（定義をしている行の行番号には#が付与されています）。

クロスリファレンス・リスティングファイルの省略時の拡張子は、PRN です。この、PRN ファイルはオペレーティングシステムのコマンド（COPY、TYPE コマンド）を使用してプリンタへ出力したり、コンソールのスクリーンへ表示させることができます。また、CREF-80 は MSX・M-80 と同じ出力装置の指定をサポートします。

6.3 クロスリファレンス・ファイルの作成

クロスリファレンス・ファイルは、MSX・M-80 でプログラムをアセンブル時に MSX・M-80 コマンド列へ MSX・M-80 スイッチの /C を指定して作成します。

例) M80 = NEIL/C

上の例は NEIL . MAC をアセンブルし、オブジェクトファイル (NEIL . REL) とクロスリファレンス・ファイル (NEIL . CRF) を生成します。

6.4 CREF-80の呼び出し

CREF-80 クロスリファレンサはコマンドで呼び出します。また、クロスリファレンサの実行時に使うファイル名を CREF-80 コマンドで指定します。CREF-80 コマンドの指定の方法は次の 2 通りがあります。コマンドは小文字で入力しても大文字で入力しても同じです。小文字は大文字に変換されます。

- (1) クロスリファレンサを呼び出すコマンドに CREF-80 コマンドを指定する。

A > CREF80 <CREF-80 コマンド>

例) A > CREF80 PRN = NEIL

- (2) クロスリファレンサを呼び出すコマンドには CREF-80 コマンドを指定せず、クロスリファレンサのプロンプト (*) に応じて指定する。

```
A > CREF80
* <CREF-80 コマンド>
```

次の例は(1)と同じ結果になります。

```
例) A > CREF80
* PRN = NEIL
```

CREF-80 は CREF-80 コマンドを処理するとアスタリスク (*) プロンプトを表示しますので、次の CREF-80 コマンドを指定することができます。

CREF-80 からオペレーティングシステムに抜け出るには **CONTROL**+**C** をキーインします。

6.5 CREF-80コマンド

6.5.1 CREF-80 コマンドの形式

コマンドの形式は次のとおりです。

〈クロスリファレンス・リスティングファイル名〉 = 〈クロスリファレンス・ファイル名〉

6.5.2 ファイル名の指定

ファイル名の指定方法は次のとおりです。

〈デバイス名〉: 〈ファイル名〉. 〈拡張子〉

例) A: NEIL . CRF

ファイル名の指定は省略することができます。また、クロスリファレンス・リスティングファイルについては論理デバイス名を指定することができます。以下にその方法を説明します。

- (1) クロスリファレンス・ファイル

クロスリファレンス・ファイルの指定は全部を省略することはできません。しかし、デバイス名と拡張子を省略することができます。デバイス名の省略時にはデフォルトドライブを、また、拡張子の省略時には .CRF が仮定されます。

(2) クロスリファレンス・リスティングファイル

クロスリファレンス・リスティングファイルは指定を全部省略すると、クロスリファレンス・ファイルと同じディスク上に拡張子が.CRFである以外クロスリファレンス・ファイルと同じファイル名で作成されます。また、デバイス名を省略するとデフォルトドライブが仮定され、ファイル名を省略するとクロスリファレンス・ファイルと同じファイル名が仮定されます。また、拡張子を省略すると.PRNが仮定されます。クロスリファレンス・リスティングファイルと異なるドライブに作成する場合、デバイス名を省略することはできません。

例) CREF80 B : = A : NEIL

また、.PRN 以外の拡張子で作成する場合拡張子を省略することはできません。

例) CREF80 NEIL . CRL = NEIL

クロスリファレンス・リスティングファイルに次のような論理デバイス名を指定してプリンタやコンソールのスクリーンへ出力することができます。

論理デバイス名	出力装置
CON	コンソール
PRN	プリンタ

6.5.3 CREF-80 コマンドの例

- クロスリファレンス・リストをプリンタへ出力する。

例) CREF80 PRN = NEIL

NEIL . CRF を入力し、クロスリファレンス・リストをプリンタへ出力します（ディスクファイルは作成されません）。

- クロスリファレンス・リストをコンソールのスクリーンへ出力する。

例) CREF80 CON = NEIL

NEIL . CRF を入力し、クロスリファレンス・リストをスクリーンへ出力します（ディスクファイルは作成されません）。

- クロスリファレンス・リスティングファイルをディスクへ出力します。

例) CREF80 NEIL = NEIL

CREF80 = NEIL

NEIL . CRF を入力し、クロスリファレンス・リスティングファイルを NEIL . LST という名前で NEIL.CRF と同じディスク上に作成します。

6.6 リスティング制御擬似命令

クロスリファレンス・リストをオプションでプログラムの全部ではなく一部分だけ出力することができます。クロスリファレンス・リストの出力・抑止を制御する場合は、MSX・M-80 の入力となるソースプログラム中でつぎに掲げるクロスリファレンス・リスティング制御擬似命令を使用してください。

これらの擬似命令はオペレータフィールドに入力します。他のリスティング制御擬似命令と同様にアークメントフィールドの指定はありません。また、これらの擬似命令はプログラム中の任意の場所に置くことができます。詳細は 3.6 の CREF 情報出力制御擬似命令の項を参照してください。

・ CREF

クロスリファレンスを作成します。

・ CREF はデフォルト状態です。・ CREF は、・ XCREF 擬似命令を使ってクロスリファレンスを抑止した後、抑止を解除してクロスリファレンスの作成を再開するときに使用します。

・ CREF の指定は・ XCREF の指定をするまで有効です。しかし、MSX・M-80 コマンドで／C スイッチを指定しないとクロスリファレンス・ファイルは作成されません。

・ XCREF

クロスリファレンスを抑止します。

・ XCREF は、・ CREF 擬似命令でセットした状態（デフォルト）を解除します。

・ XCREF の効果は、・ CREF の指定をするまで有効です。

プログラムの一部でクロスリファレンスの生成を抑止するときに・ XCREF を使用してください。

・ CREF と・ XCREF は MSX・M-80 コマンドで／C スイッチを指定しないと効果はありませんから、通常のリスティングファイル（クロスリファレンス以外のもの）を出力させたい場合でも・ XCREF を使う必要はなく MSX・M-80 コマンドに／C スイッチを指定しただけでよい。

6.7 リスティングフォーマット

クロスリファレンス・リストの例を次に示します。

```
SOME USEFUL SUBROUTINES MSX-M-80 1.00 01-Apr-85 PAGE 1

①→ 1          TITLE SOME USEFUL SUBROUTINES
2          ENTRY MUL10,DMUL10,DIV10
3
4          ; MUL10 - [A]=A^H[A]*10, [B] IS DESTROYED
5          ;
6          MUL10: MOV     B,A
7                  RLC
8                  RLC
9                  ADD     B
10                 RLC
11                 RET      ;A=(A*2*2+A)*2
12
13          ; DMUL10 - [HL]=[A]*10
14          ;
15          DMUL10: PUSH   B
16                  MOV    C,A
17                  MVI    B,0
18                  MOV    L,A
19                  MVI    H,0      ;SET UP [BC] AND [HL]
20                  DAD    H
21                  DAD    H
22                  DAD    B
23                  DAD    H      ;[HL]=([HL]*2*2+[BC])*2
24                  POP    B
25                  RET
26
27          ; DIVIDE BY 10
28          ; B=A/10 A=A MOD 10
29          ;
30          DIV10: MVI     B,0      ;CLR QUOT.
31                  INR     B
32          DVLOP: SUI     10      ;TRY SUB
33                  JNC     DVLOP   ;TRY OK, THEN ANOTHER TRY
34                  DCR     B      ;OOPS! EXCEED 0.
35                  ADI     10      ; THEN ADJUST STUFF
36                  RET
37          END      ; & RETURN
```

SOME USEFUL SUBROUTINES MSX-M-80 1.00 01-Apr-85 PAGE 5

Macros:

Symbols:

```
0013I' DIV10          0006I' DMUL10          0015' DVLOP
0000I' MUL10
```

No Fatal error(s)

```
②→ DIV10      2      30#
   DMUL10      2      15#
   DVLOP      31#     33
   MUL10       2       6#
```

①クロスリファレンス番号

②変数名と変数名が定義してある行と参照している行のクロスリファレンス番号



第 7 章

LIB-80ライブラリマネージャ

- 7.1 LIB-80
- 7.2 LIB-80で使用するファイル
- 7.3 LIB-80の呼び出し
- 7.4 LIB-80コマンド
- 7.5 スイッチ
- 7.6 注意事項

第7章

LIB-80 ライブラリマネージャ

7.1 LIB-80

LIB-80 ライブラリマネージャは、オブジェクトモジュールを集めたライブラリファイル进行管理するユーティリティプログラムで、MSX-C コンパイラのライブラリファイルを作成することができます。LIB-80 の機能は次のとおりです。

- オブジェクトファイル、あるいは既存のライブラリファイルに含まれるオブジェクトモジュールを集めて、新しくライブラリファイルを作成します。
- ライブラリファイルに含まれるオブジェクトモジュール名と、オブジェクトモジュールに含まれているパブリックシンボルの参照、被参照状態をコンソールのスクリーンに表示します。

LIB-80 を使ってオブジェクトモジュールをライブラリファイルに集めておくと、LIB-80 でコマンドに〈ライブラリファイル名〉/S を指定するだけで必要なオブジェクトモジュールがすべて実行可能ファイルにリンクされます。この機能はサブルーチンが多いプログラムの場合、LINK-80 コマンドで1つ1つオブジェクトモジュールファイルを指定するのに較べて大変便利です。

注意 LIB-80 は新たなライブラリファイルを作成するのに便利な反面、コマンドの指示を誤ると既存のライブラリファイルを破壊する可能性があります。MSX・L-80 の使用前に既存のライブラリファイルのバックアップコピーを取っておいてください。

7.2 LIB-80で使用するファイル

LIB-80 ライブラリマネージャの実行時に使用するファイルは次のとおりです。

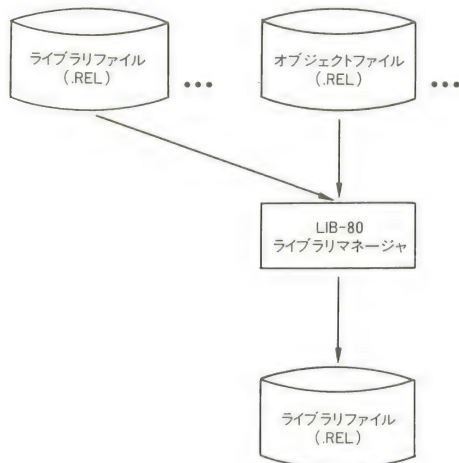


図 7-1 LIB-80で使用するファイル

(1) オブジェクトファイル

END<式>あるいは ENTRY ステートメントを含むアセンブリ言語のプログラムを MSX・M-80 でアセンブルした結果、または FORTRAN-80, COBOL-80, BASIC などのコンパイル結果であるオブジェクトファイルです。

(2) ライブラリファイル

オブジェクトファイルを LIB-80 で集めたライブラリイメージのファイルです。

7.3 LIB-80の呼び出し

LIB-80 ライブラリマネージャはコマンドで呼び出します。また、LIB-80 の実行時に使うファイル名やスイッチをLIB-80 コマンドで指定します。

LIB-80 コマンドの指定の方法は次の2通りあります。

- (1) ライブラリマネージャを呼び出すコマンドに LIB-80 コマンドを指定します。

```
A > LIB80 <LIB-80 コマンド>
```

例) A > LIB80 <LIB-80 コマンド>

この場合、LIB-80 コマンドを小文字で入力しても大文字に変換されます。

- (2) ライブラリマネージャを呼び出すコマンドに LIB-80 コマンドを指定せず、ライブラリマネージャのプロンプト (*) に応じて指定します。

```
A > LIB80
* > <LIB-80 コマンド>
:
```

この方法では LIB-80 コマンドをいくつかに分けて入力することができます。

LIB-80 からオペレーティングシステムへ抜け出す場合には **CONTROL**+**C** をキーインします。

7.4 LIB-80コマンド

7.4.1 LIB-80 コマンドの形式

コマンドの形式は次のとおりです。

[<宛先フィールド> =] <ソースフィールド> <スイッチ>

宛先フィールド

宛先フィールドの指定はオプションです。=は宛先フィールドの指定をする場合に必要となります。宛先フィールドには LIB-80 で作成するライブラリファイル名を指定します。

<ライブラリファイル名>

このフィールドを省略すると FORLIB . REL (FORLIB . LIB ではありません) が仮定されます。このため、デフォルトドライブ上に提供された FORLIB . REL ランタイムライブラリがあると FORLIB . REL が書き換わります。

FORLIB . REL を新しく作成しない限り、宛先フィールドの指定は指定してください。

ソースフィールド

ソースフィールドの指定は省略することはできません。ソースフィールドには宛先フィールドに指定したライブラリファイルへ加えたいオブジェクトファイル名、または既に作成済みのライブラリファイルに含まれているオブジェクトモジュール名です。

<div> <div> <オブジェクトファイル名> <ライブラリファイル名> <<オブジェクトモジュール名>> <ライブラリファイル名> </div> <div>[</div> <div> <オブジェクトファイル名> <ライブラリファイル名><<オブジェクトモジュール名>> <ライブラリファイル名> </div> <div>...]</div> </div>
--

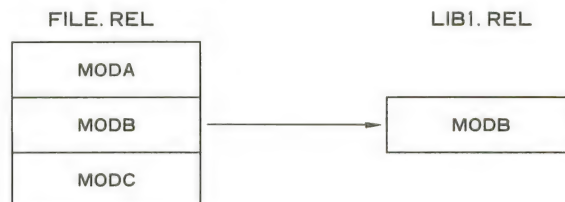
- オブジェクトファイル名やライブラリファイルに含まれているオブジェクトモジュール名はカンマ(,)で区切っていくつでも指定できます。
- ライブラリファイルに含まれているオブジェクトモジュール名は山形かっこ(<>)で囲みます。

例) * FILE1, FILE2 <MODZ>, FILE3 <MODR>, FILE4

- ライブラリファイルに含まれているオブジェクトモジュールは次のように指定することができます。

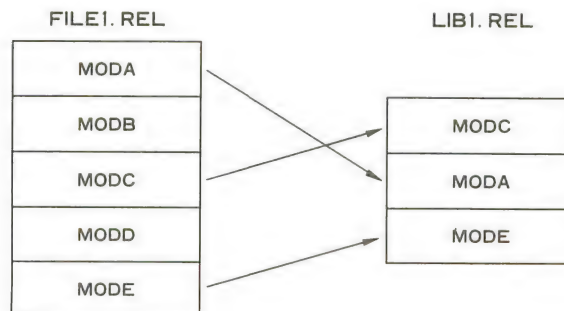
a. ライブラリに含まれているオブジェクトモジュールを1つだけ抜き出す。

例) * LIB1 = FILE1 <MODB>



b. ライブラリファイルに含まれているオブジェクトモジュールをカンマで区切っていくつか抜き出す。

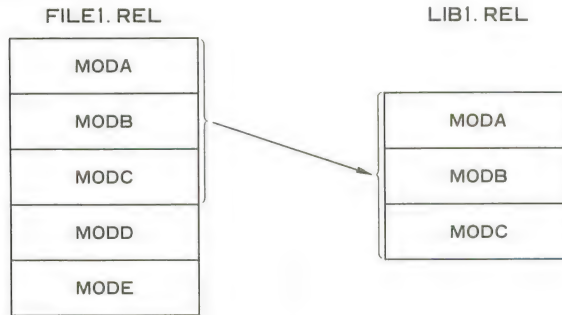
例) * LIB1 = FILE1 <MODC, MODA, MODE>



オブジェクトモジュールは既存のライブラリファイルでの順序とは関係なく、指定した順序で新しいライブラリファイルへ入ります。

- c. ライブラリファイルの先頭から、指定したオブジェクトモジュールまでを抜き出す。

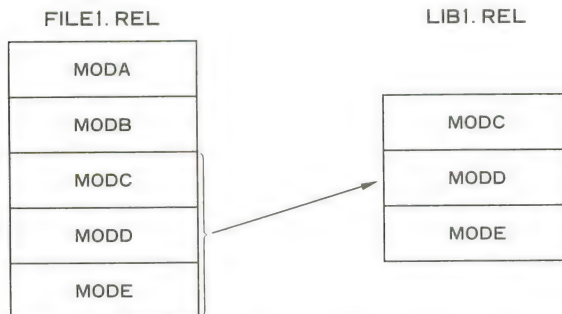
例) * LIB1 = FILE1 <., MODC>



抜き出す最後のオブジェクトモジュール名の前にはピリオドを2つ付けます。

- d. 指定したオブジェクトモジュールからライブラリファイルの最後までを抜き出す。

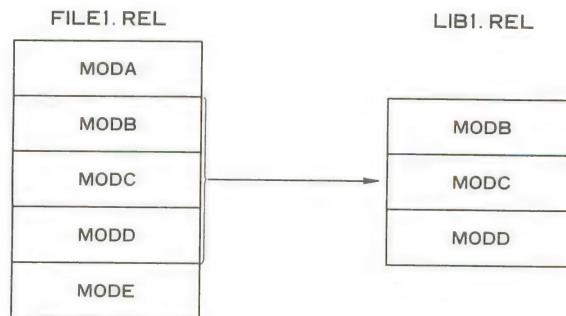
例) * LIB1 = FILE1 <MODC,.>



抜き出す先頭のオブジェクトモジュール名の後にはピリオドを2つ付けます。

- e. 指定したオブジェクトモジュールから別に指定したオブジェクトモジュールまでを連続して抜き出す。

例) * LIB1 = FILE1 <MODB., MODD>

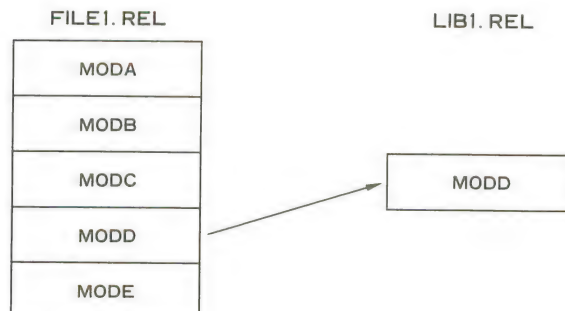


抜き出す先頭のオブジェクトモジュール名と最後のオブジェクトモジュール名の間にはピリオドを2つ入れます。

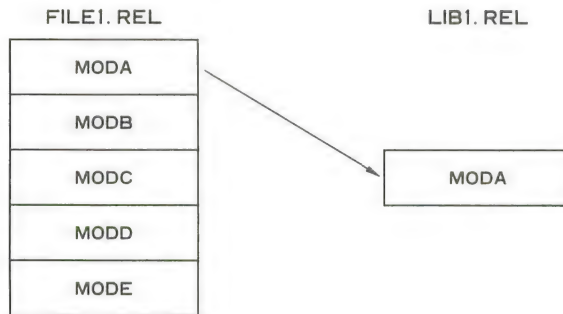
- f. 抜き出すオブジェクトモジュールの指定をオブジェクトモジュール名+オフセット番号, オブジェクトファイル名-オフセット番号で指定する。

オフセット番号は1~255です。

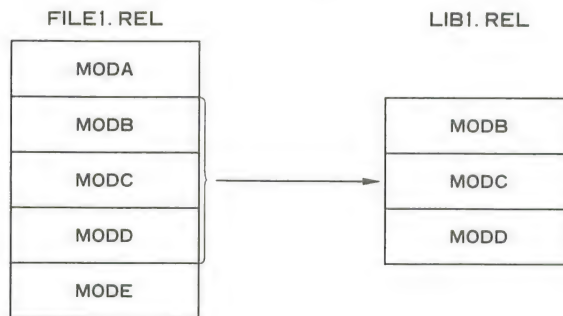
例) * LIB1 = FILE1 <MODB + 2>



例) * LIB1 = FILE1 <MODD-3>

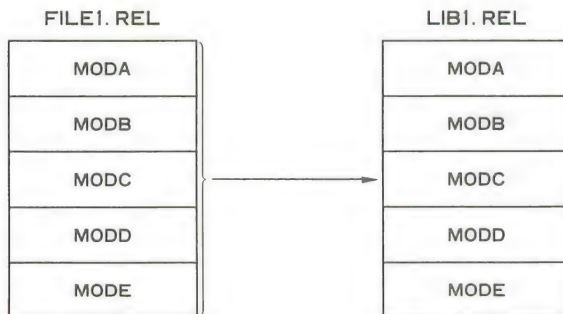


例) * LIB1 = FILE1 <MODA + 1, MODE - 1>



g. ライブラリファイルに含まれているオブジェクトモジュールを全部抜き出す。

例) * LIB1 = FILE1



7.4.2 ファイル名の指定

ファイル名の指定方法は次のとおりです。

〈デバイス名〉：〈ファイル名〉．〈拡張子〉

例) A : NEIL . REL

ファイル名の指定方法の詳細は MSX・L-80 で説明したものと同じです。

(1) 宛先フィールド

宛先フィールドのライブラリファイル名の指定はオプションですが、省略をするとデフォルトドライブ上の FORLIB . REL が仮定されますので注意して下さい。また、デバイス名の省略時にはデフォルトドライブ、拡張子の省略時に、REL がそれぞれ仮定されます。

(2) ソースフィールド

ソースフィールドにはライブラリファイル名かオブジェクトファイル名を1つ以上指定します。デバイス名の省略時にはデフォルトドライブ、拡張子の省略時には、REL がそれぞれ仮定されます。

7.4.3 LIB-80 コマンドの例

- オブジェクトモジュールを集めてライブラリファイルを作成する。

例) A > LIB80

* TRANLIB = SIN, COS, TAN, ACOG

* OLDLIB <EXP>

* /E

SIN . REL, COS . REL, TAN . REL, ACOG . REL および OLDLIB . REL に含まれる EXP モジュールを集めて TRANLIB . REL というライブラリファイルを作成します。

- ライブラリファイルのリスティングを表示する。

例) A > LIB80

* TRANLIB . LIB/U

* TRANLIB . LIB/L

TRANLIB . LIB に含まれているパブリックシンボルの内 MSX・M-80 で未定義となる可能性のあるシンボルのリスティングと、TRANLIB . LIB に含まれているすべてのモジュールとパブリックシンボルのクロスリファレンス・リスティングとをコンソールのスクリーンに表示します。

7.5 スイッチ

スイッチは LIB-80 に対して追加機能の実行を指定します。スイッチはスラッシュ(/)が付いた文字で、スイッチフィールドへいくつでも指定することができます。スイッチの一覧表を次に示します。

表 7-1 スイッチ一覧

スイッチ	説 明
/E	現在作成中のライブラリを宛先フィールドで指定したファイル名でセーブしオペレーティングシステムへ抜け出ます。ライブラリファイルと同じ名前のファイルがあれば既存のファイルが削除されます。
/R	現在作成中のライブラリを宛先フィールドで指定したファイル名でセーブします。オペレーティングシステムには抜け出しません。ライブラリファイルと同じ名前のファイルがあれば既存のファイルが削除されます。現在作成中のライブラリファイルをセーブ後、他のライブラリファイルを作成したい場合に使用します。
/L	指定されたファイル内のオブジェクトモジュール名と、オブジェクトモジュールに含まれているグローバルシンボル定義をクロスリファレンス形式でコンソールのスクリーンへ表示します。プリンタへ出力したい場合はLIB-80実行前に CONTROL + P をキーインしてください。
/U	ライブラリファイル中のオブジェクトモジュールに含まれるグローバルシンボルのうち、未定義となる可能性のあるものをコンソールのスクリーンへリストします。
/C	現在作成中のライブラリをクリアし、LIB-80を初期状態に戻します。LIB-80のアスタリスク(*)プロンプトが表示され新しいLIB-80コマンドを待ちます。 /CはLIB-80コマンドを間違えてキーインした場合などに使用してください。
/O	/Lスイッチを指定して出力するリスト中の数値の基数を8進にします。 /Oは/Lスイッチと共に指定します。 例 NEWLIB/L/O
/H	/Lスイッチを指定して出力するリスト中の数値の基数を16進にします。16進はデフォルト値ですので、/Hを指定しなくても自動的に16進基数となります。

7.6 注意事項

ライブラリファイルはリンクロード時にロード済みのオブジェクトモジュールだけでは解決できないグローバルリファレンスがあったとき、1パスだけ検索されます。このため、ライブラリファイルへオブジェクトモジュールを集める場合、他のモジュールから参照されるエントリーポイントを持つモジュールが後へ配置されるように LIB-80 コマンドで指示してください。

付 録

- A. ASCIIキャラクターコード表
- B. MSX・M-80擬似命令表
- C. オペコード一覧表
- D. オブジェクトファイルの形成

付録A

ASCIIキャラクタコード表

10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ	10進	16進	キャラクタ
000	00 H	NUL	032	20 H	SPACE	064	40 H	@	096	60 H	`
001	01 H	SOH	033	21 H	!	065	41 H	A	097	61 H	a
002	02 H	STX	034	22 H	"	066	42 H	B	098	62 H	b
003	03 H	ETX	035	23 H	#	067	43 H	C	099	63 H	c
004	04 H	EOT	036	24 H	\$	068	44 H	D	100	64 H	d
005	05 H	ENQ	037	25 H	%	069	45 H	E	101	65 H	e
006	06 H	ACK	038	26 H	&	070	46 H	F	102	66 H	f
007	07 H	BEL	039	27 H	'	071	47 H	G	103	67 H	g
008	08 H	BS	040	28 H	(072	48 H	H	104	68 H	h
009	09 H	HT	041	29 H)	073	49 H	I	105	69 H	i
010	0A H	LF	042	2A H	*	074	4A H	J	106	6A H	j
011	0B H	VT	043	2B H	+	075	4B H	K	107	6B H	k
012	0C H	FF	044	2C H	,	076	4C H	L	108	6C H	l
013	0D H	CR	045	2D H	-	077	4D H	M	109	6D H	m
014	0E H	SO	046	2E H	.	078	4E H	N	110	6E H	n
015	0F H	SI	047	2F H	/	079	4F H	O	111	6F H	o
016	10 H	DLE	048	30 H	0	080	50 H	P	112	70 H	p
017	11 H	DC1	049	31 H	1	081	51 H	Q	113	71 H	q
018	12 H	DC2	050	32 H	2	082	52 H	R	114	72 H	r
019	13 H	DC3	051	33 H	3	083	53 H	S	115	73 H	s
020	14 H	DC4	052	34 H	4	084	54 H	T	116	74 H	t
021	15 H	NAK	053	35 H	5	085	55 H	U	117	75 H	u
022	16 H	SYN	054	36 H	6	086	56 H	V	118	76 H	v
023	17 H	ETB	055	37 H	7	087	57 H	W	119	77 H	w
024	18 H	CAN	056	38 H	8	088	58 H	X	120	78 H	x
025	19 H	EM	057	39 H	9	089	59 H	Y	121	79 H	y
026	1A H	SUB	058	3A H	:	090	5A H	Z	122	7A H	z
027	1B H	ESCAPE	059	3B H	;	091	5B H	[123	7B H	{
028	1C H	FS	060	3C H	<	092	5C H	\	124	7C H	
029	1D H	GS	061	3D H	=	093	5D H]	125	7D H	}
030	1E H	RS	062	3E H	>	094	5E H	^	126	7E H	~
031	1F H	US	063	3F H	?	095	5F H	_	127	7F H	DEL

LF=ライン・フィード、FF=フォーム・フィード、CR=キャリッジ・リターン、DEL=ラフアウト

付録B

MSX・M-80擬似命令表

MSX・M-80 の擬似命令を機能ごとの一覧表に示します。なお、SET 擬似命令は Z80 モードでは使用できません。SET 以外の擬似命令は Z80 モードでも、8080 モードでも使用できます。

命令セットの選択

擬似命令	説 明
.Z80	Z80 命令セットモードにする
.8080	8080 命令セットモードにする

データ定義とシンボル定義

擬似命令			説 明
<シンボル>	ASET	<式>	<式>の値にシンボルを割り当て
[<ラベル>]	BYTE EXT	<シンボル>	エクスターナルシンボル宣言
[<ラベル>]	BYTE EXTRN	<シンボル>	エクスターナルシンボル宣言
[<ラベル>]	BYTE EXTERNAL	<シンボル>	エクスターナルシンボル宣言
[<ラベル>]	DB	<式>[, <式>...]	1バイトずつの領域を確保し、初期値をセット
[<ラベル>]	DB	<ストリング>[<ストリング>...]	1バイトずつの領域を確保し、初期値をセット
[<ラベル>]	DC	<ストリング>	ストリングの領域を確保し、初期値をセット
[<ラベル>]	DEFB	<式>[, <式>...]	1バイトずつの領域を確保し、初期値をセット
<シンボル>	DEFL	<式>	<式>の値にシンボルを割り当て
[<ラベル>]	DEFM	<ストリング>[, <ストリング>...]	1バイトずつの領域を確保し、初期値をセット
[<ラベル>]	DEFS	<式>[, <値>]	領域を確保し、初期値をセット
[<ラベル>]	DEFW	<式>[, <式>...]	2バイトのワード領域を確保し、初期値をセット
[<ラベル>]	DS	<式>[, <値>]	領域を確保し、初期値をセット
[<ラベル>]	DW	<式>[, <式>...]	2バイトのワード領域を確保し、初期値をセット
	ENTRY	<シンボル>[, <シンボル>...]	パブリックシンボル宣言
<シンボル>	EQU	<式>	<シンボル>に<式>の値を割り当て
	EXT	<シンボル>[, <シンボル>...]	外部参照宣言
	EXTRN	<シンボル>[, <シンボル>...]	外部参照宣言
	EXTERNAL	<シンボル>[, <シンボル>...]	外部参照宣言
	GLOBAL	<シンボル>[, <シンボル>...]	パブリックシンボル宣言
	PUBLIC	<シンボル>[, <シンボル>...]	パブリックシンボル宣言
<シンボル>	SET	<式>	<式>の値にシンボルを割り当て

PCモード指定

擬似命令	説 明
ASEG	絶対モードにする
CSEG	コード相対モードにする
DSEG	データ相対モードにする
COMMON /<ブロック名>/	COMMON相対モードにする
ORG <式>	ロケーションカウンタの値を変更
.PHASE <式>	.PHASEから.DEPHASEまでのプログラムを<式>で
.DEPHASE	指定したアドレスで実行

ファイル関係

擬似命令	説 明
.COMMENT <区切り記号><テキスト><区切り記号>	<区切り記号>ではさまれた<テキスト>をコメントとする
END [<式>]	プログラムの終了
INCLUDE <ファイル名>	他のファイルに含まれるステートメントを論理的に挿入
\$INCLUDE <ファイル名>	他のファイルに含まれるステートメントを論理的に挿入
MACLIB <ファイル名>	他のファイルに含まれるステートメントを論理的に挿入
NAME (^<モジュール名>^)	モジュール名定義
.RADIX <式>	基数の指定
.REQUEST <ファイル名>[<ファイル名>...]	外部モジュールのリンクロードを要求

リスティングフォーマット制御

擬似命令	説 明
PAGE <式>	リスティングの改ページ
SUBTTL <テキスト>	サブタイトルの指定
TITLE <テキスト>	タイトルの指定
\$TITLE (^<テキスト>^)	サブタイトルの指定

一般リスティング制御

擬似命令	説 明
.LIST	リスティングを出力
.XLIST	リスティングの出力を抑止
.PRINTX <区切り記号><テキスト><区切り記号>	アセンブル中にコンソールメッセージを出力する

条件リスティング制御

擬似命令	説 明
.LFCOND	偽条件ブロックのリスティングを出力
.SFCOND	偽条件ブロックのリスティングの出力を抑止
.TFCOND	／Xスイッチの有無により偽条件ブロックのリスティングの出力を決定する

マクロ拡張リスティング制御

擬似命令	説 明
.LALL	マクロ、リピートブロックの展開部分をリスティングする
.XALL	マクロ、リピートブロックの展開部分をリスティングしない
.SALL	マクロ、リピートブロックの展開部分のうち、オブジェクトコードを生成するステートメントをリスティングする

クロスリファレンス情報出力制御

擬似命令	説 明
.CREF	クロスリファレンス情報を出力
.XCREF	クロスリファレンス情報の出力を抑止

マクロ

擬似命令	説 明
<名前> MACRO [<パラメータ>[,<パラメータ>…]]	マクロ定義を開始
ENDM	マクロ定義を終了
EXITM	マクロの展開を終了
LOCAL <ダミー>[,<ダミー>…]	マクロ展開時にマクロ定義中のダミーパラメータをユニークなシンボルに置き換える

反復

擬似命令	説 明
REPT <式>	反復ブロックを指定した数だけくり返し展開
IRP <ダミー>,<山形カッコで囲まれたパラメータ>	反復ブロックを指定したパラメータに従ってくり返し展開
IRPC <ダミー>,<文字列>	反復ブロックを指定した文字列に従ってくり返し展開

条件付きアセンブリ機能

擬似命令	説 明
COND <式>	<式>がゼロ以外の値のとき条件は真とみなされ、条件ブロック内がアセンブルされる
ELSE	IF××××や CONDの条件が偽となったときに展開するステートメントを開始する
ENDC	.CONDで開始した条件ブロックを終了する
ENDIF	IF××××で開始した条件ブロックを終了する
IF <式>	<式>がゼロ以外の値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFB <アーギュメント>	<アーギュメント>がブランクのとき条件は真とみなされ条件ブロック内がアセンブルされる
IFDEF <シンボル>	<シンボル>が定義されているとき条件は真とみなされ、条件ブロック内がアセンブルされる
IFDIF <アーギュメント 1>,<アーギュメント 2>	<アーギュメント 1>と<アーギュメント 2>が同一のとき条件は真とみなされ、条件ブロック内がアセンブルされる
IFE <式>	<式>がゼロの値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFF <式>	<式>がゼロの値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFIDN <アーギュメント 1>,<アーギュメント 2>	<アーギュメント 1>と<アーギュメント 2>が同一のとき条件は真とみなされ条件ブロック内がアセンブルされる
IFNB <アーギュメント>	<アーギュメント>がブランクでないとき条件は真とみなされ、条件ブロック内がアセンブルされる
IFNDEF <シンボル>	<シンボル>が定義されていないとき条件は真とみなされ、条件ブロック内がアセンブルされる
IFT <式>	<式>がゼロ以外の値のとき条件は真とみなされ条件ブロック内がアセンブルされる
IF1	アセンブラがパス 1 を実行中、条件が真とみなされ条件ブロック内がアセンブルされる
IF2	アセンブラがパス 2 を実行中条件が真とみなされ条件ブロック内がアセンブルされる

付録C

オペコード一覧表

Z 80 オペコード一覧表と、8080 オペコード一覧表を示します。

Z80 オペコード一覧

オペコード		機 能
ADC	A	Add with Carry to Accumulator
ADC	HL, rp	Add Register Pair with Carry to HL
ADD		Add
AND		Logical AND
BIT		Test Bit
CALL	Addr	Call Subroutine
CALL	cond, addr	Call Conditional
CCF		Complement Carry Flag
CP		Compare
CPD		Compare, Decrement
CPDR		Compare, Decrement, Repeat
CPI		Compare, Increment
CPIR		Compare, Increment, Repeat
CPL		Complement Accumulator
DAA		Decimal Adjust Accumulator
DEC		Decrement
DI		Disable Interrupts
DJNZ		Decrement and Jump if Not Zero
EI		Enable Interrupts
EX		Exchange
EXX		Exchange Register Pairs and Alternatives
HALT		Halt
IM	x	Set Interrupt Mode
IN		Input

INC		Increment
IND		Input, Decrement
INDR		Input, Decrement, Repeat
INI		Input, Increment
INIR		Input, Increment, Repeat
JP	addr	Jump
JP	cond, addr	Jump Conditional
JR		Jump Relative
JR	cond, addr	Jump Relative Conditional
LD	A, (addr)	Load Accumulator Direct
LD	A, (BC) or (DE)	Load Accumulator Secondary
LD	A, I	Load Accumulator from Interrupt Vector Register
LD	A, R	Load Accumulator from Refresh Register
LD	HL, (addr)	Load HL Direct
LD	data	Load Immediate
LD	xy, (addr)	Load Index Register Direct
LD	reg, (HL)	Load Register
LD	reg, (xy + disp)	Load Register Indexed
LD	rp, (addr)	Load Register Pair Direct
LD	SP, HL	Move HL to Stack Pointer
LD	SP, xy	Move Index Register to Stack Pointer
LD	dst, scr	Move Register-to-Register
LD	(addr) , A	Store Accumulator Direct
LD	(BC) or (DE) , A	Store Accumulator Secondary
LD	I, A	Store Accumulator to Interrupt Vector Register
LD	R, A	Store Accumulator to Refresh Register
LD	(addr) , HL	Store HL Direct
LD	(HL) , data	Store Immediate to Memory
LD	(xy + disp) , data	Store Immediate to Memory Indexed
LD	(addr) , xy	Store Index Register Direct
LD	(HL) , reg	Store Register
LD	(xy + disp) , reg	Store Register Indexed
LD	(addr) , rp	Store Register Pair Direct
LDD		Load, Decrement
LDDR		Load, Decrement, Repeat
LDI		Load, Increment
LDIR		Load, Increment, Repeat

NEG		Negate (Two's Complement) Accumulator
NOP		No Operation
OR		Logical OR
OUT		Output
OUTD		Output, Decrement
OTDR		Output, Decrement, Repeat
OUTI		Output, Increment
OTIR		Output, Increment, Repeat
POP		Pop from Stack
PUSH		Push to Stack
RES		Reset Bit
RET		Return from Subroutine
RET	cond	Return Conditional
RETI		Return from Interrupt
SETN		Return from Non-Maskable Interrupt
RL		Rotate Left Through Carry
RLA		Rotate Accumulator Left Through Carry
RLC		Rotate Left Circular
RLCA		Rotate Accumulator Left Circular
RLD		Rotate Accumulator and Memory Left Decimal
RR		Rotate Right Through Carry
RRA		Rotate Accumulator Right Through Carry
RRC		Rotate Right Circular
RRCA		Rotate Accumulator Right Circular
RRD		Rotate Accumulator and Memory Right Decimal
RST		Restart
SET		Set Bit
SBC		Subtract with Carry (Borrow)
SCF		Set Carry Flag
SLA		Shift Left Arithmetic
SRA		Shift Right Arithmetic
SRL		Shift Right Logical
SUB		Subtract
XOR		Logical Exclusive OR

8080 オペコード一覧

オペコード	機 能
ADC, ACI	Add with Carry
ADD, ADI	Add
ANA, ANI	Logical AND
CALL	Call Subroutine
CC	Call on Carry
CM	Call on Minus
CMA	Complement Accumulator
CMC	Complement Carry
CMP, CPI	Compare
CNC	Call on No Carry
CNZ	Call on Not Zero
CP	Call on Positive
CPE	Call on Parity Even
CPO	Call on Parity Odd
CZ	Call on Zero
DAA	Decimal Adjust
DAD	16-bit Add
DCR	Decrement
DCX	16-bit Decrement
DI	Disable Interrupts
EI	Enable Interrupts
HLT	Halt
IN	Input
INR	Increment
INX	Increment 16 bits
JC	Jump on Carry
JM	Jump on Minus
JMP	Jump
JNC	Jump on Not Carry
JNZ	Jump on Not Zero
JP	Jump on Positive
JPE	Jump on Parity Even
JPO	Jump on Parity Odd

JZ	Jump on Zero
LDA	Load Accumulator
LDAX	Load Accumulator Indirect
LHLD	Load HL Direct
LXI	Load 16 bits
MOV	Move
MVI	Move Immediate
NOP	No Operation
ORA, ORI	Logical OR
OUT	Output
PCHL	HL to Program Counter
POP	Pop from Stack
PUSH	Push to Stack
RAL	Rotate with Carry Left
RAR	Rotate with Carry Right
RC	Return on Carry
RET	Return from Subroutine
RLC	Rotate Left
RM	Return on Minus
RNC	Return on No Carry
RNZ	Return on Not Zero
RP	Return on Positive
RPE	Return on Parity Even
RPO	Return on Parity Odd
RRC	Rotate Right
RST	Restart
RZ	Return on Zero
SBB, SBI	Subtract with Borrow
SHLD	Store HL Direct
SPHL	HL to Stack Pointer
STA	Store Accumulator
STAX	Store Accumulator Indirect
STC	Set Carry
SUB, SUI	Subtract
XCHG	Exchange D and E, H and L
XRA, XRI	Logical Exclusive OR
XTHL	Exchange Top of Stack, HL

付録D

オブジェクトファイルの形式

この付録は MSX・L-80 のリロケートブルなオブジェクトファイルのロード形式を知りたい方のための参考資料です。特に必要がなければ読みとばしてください。

オブジェクトファイルはビットストリームより成ります。ビットストリームの中の個々のフィールドは次に述べるものを除いてバイト境界に調整されません。リロケートブルなオブジェクトファイルにビットストリームを用いる事によりオブジェクトファイルのサイズが小さくなり、ディスクに対する読み出し、書き込みの回数を減らす事ができます。

ロードアイテムにはアブソリュートとリロケートブルアイテムとの2つの基本的な型があります。ロードアイテムの最初のビットが0であればアブソリュートアイテムであり、また、ビットが1であればリロケートブルアイテムです。さらに、リロケートブルアイテムでは次の2ビットにより4つの型に分けられます。

MSX・L-80 ではオブジェクトファイル中のロードアイテムの種類によって次の表のような編集を行なってリンクロードします。

表 D-1 ロードアイテム一覧

ロードアイテム	説 明
アブソリュートなロードアイテム <div> <div>0</div> <div>アブソリュートバイト</div> </div> <div>0 1 9</div>	2ビット目以降の8ビットはアブソリュートバイトとしてロードされる。
特別なLINKアイテム <div> <div>1</div> <div>0 0</div> <div>制御フィールド</div> </div> <div>0 1 3 7</div> <div> <div>1</div> <div>0 0</div> <div>0 0 0 0</div> <div>Bフィールド</div> </div> <div> <div>1</div> <div>0 0</div> <div>0 0 0 1</div> <div>Bフィールド</div> </div> <div> <div>1</div> <div>0 0</div> <div>0 0 1 0</div> <div>Bフィールド</div> </div> <div> <div>1</div> <div>0 0</div> <div>0 0 1 1</div> <div>Bフィールド</div> </div> <div> <div>1</div> <div>0 0</div> <div>0 1 0 0</div> <div>Cフィールド</div> </div>	特別なLINKアイテムはビットストリーム100から始まり、4ビットの制御フィールドによって次のような種類があります。 エントリ記号 (検索の対象となる名前). COMMONブロックの選択. プログラム名. ライブラリ検索の要求. 拡張子LINKアイテム.

ロードアイテム	説 明
<div>1 0 0 0 1 0 1 Aフィールド Bフィールド</div>	COMMONの大きさの定義。
<div>1 0 0 0 1 1 0 Aフィールド Bフィールド</div>	外部参照記号のチェーン (Aフィールドはチェーンの先頭、Bフィールドは外部参照記号)。
<div>1 0 0 0 1 1 1 Aフィールド Bフィールド</div>	エントリポイントの指定 (Aフィールドはアドレス、Bフィールドは名前)。
<div>1 0 0 1 0 0 0 Aフィールド</div>	外部参照記号+オフセット、外部参照記号に対するジャンプ命令のために使用される。
<div>1 0 0 1 0 0 1 Aフィールド</div>	外部参照記号+オフセット、Aフィールドの値は実行の直前に現ロケーションカウンタが示すアドレスから始まる2バイトに加えられる。
<div>1 0 0 1 0 1 0 Aフィールド</div>	データ領域の大きさを定義。 (Aフィールドは大きさ)。
<div>1 0 0 1 0 1 1 Aフィールド</div>	ローディング時のロケーションカウンタ (Aフィールドはロケーションカウンタ)。
<div>1 0 0 1 1 0 0 Aフィールド</div>	アドレスのチェーン (Aフィールドはチェーンの先頭番地であり、チェーンで結ばれているすべてのエントリはロケーションカウンタの現在の値で置き換えられる。チェーンの最後のエントリはアブソリュート番地0をアドレスフィールドとして持っている)。
<div>1 0 0 1 1 0 1 Aフィールド</div>	プログラムの大きさの定義。 (Aフィールドは大きさ)。
<div>1 0 0 1 1 1 0 Aフィールド</div>	プログラムの最後。 (バイト境界にされる)。
<div>1 0 0 1 1 1 1</div>	ファイルの終り。
プログラム相対 <div>1 0 1 16ビットの値</div> <div>0 1 3 19</div>	当該プログラムのベースアドレスを加えてから後に続く16ビットをロードする。
データ相対 <div>1 1 0 16ビットの値</div> <div>0 1 3 19</div>	当該プログラムのベースアドレスを加えてから後に続く16ビットをロードする。
COMMON相対 <div>1 1 1 16ビットの値</div> <div>0 1 3 19</div>	当該プログラムのベースアドレスを加えてから後に続く16ビットをロードする。

A フィールド :

YY	16 ビットの値
0	2
	18

YY : 2 ビットの番地型フィールド

00…絶対アドレス

01…プログラムリラティブ

10…データリラティブ

11…コモンリラティブ

B フィールド :

ZZZ	シンボル (1~6 バイトの文字列)
0	3

ZZZ=シンボルの長さ (3 ビット)

シンボル=1~6 文字の文字列

C フィールド :

III	S	bbbbbb
0	3	11

III = 追加情報の長さ (3 ビット)

F80 はブランクコモンについて 0 のエントリ長を作成しますので 0 は 1 を意味します

S = 次のような 8 ビットのサブタイプ識別子

5 (X'35')…COBOL オーバーレイセグメント標識

A (X'41')…算術フィクスアップ (算術演算子)

B (X'42')…算術フィクスアップ (外部参照)

C (X'43')…算術フィクスアップ (領域ベース+オフセット)

bbbbbb = 1~6 バイトの追加情報

名前に対する LINK アイテムの場合追加情報は 6 文字だけです。

拡張子 LINK アイテムの説明

オーバーレイセグメント標識の場合追加情報の長さは 2 バイト (III=010) であり、当該オーバーレイセグメント番号が値 b+49 にセットされます。このセグメントは直前のセグメント番号が 0 以外である場合、かつ、/N スイッチが有効である場合にファイル名が現プログラム名であり、拡張子が.Vnn であるディスク上のファイルに書き込まれます。(nn は数字 (b+49)₁₀を表す 2 文字の 16 進数です)。

サブタイプ A, B, および C によってポーランド式算術テキストの処理ができます。各アイテムは逆ポーランド式として読み取らなければなりません。1 つ以上のサブタイプ B または C のアイテムの次に 1 つ以上の算術演算子 (サブタイプ A) が続き、Store-Result 算術演算子 (B.STBT または B.STWT) で終了します。すべての項目はオフセットエントリがすべて最終 (絶対アドレスに変換されたあとフィクスアップ・テーブルに置かれます。ポーランド式はリンクの終りにフィクスアップ・テーブル外で実行され、その結果は LINK アイテムを読み込んだときの PC にストアされます。

索引

記号	
!	78
\$EJECT	55
\$INCLUDE	50
\$TITLE	56
%	79
&	76
*	26
+	27
-	27
- (負号)	26
.8080	30
.COMMENT	48
.CREF	63
.DEPHASE	46
.LALL	62
.LFCOND	60
.LIST	57
.PHASE	46
.PRINTX	59
.RADIX	52
.REQUEST	53
.SALL	62
.SFCOND	60
.TFCOND	61
.XALL	62
.XCREF	63
.XLIST	58
.Z80	30
/	26
::	77
8080 オペコード	24
A	
AND	27
ASCII キャラクタコード	140
ASCII スtring	21
ASEG	18, 41
ASET	39
B	
BYTE EXT	37
BYTE EXTERNAL	37
BYTE EXTRN	37
C	
COMMON	18, 44
COMMON 相対モード	18, 44
COND~ELSE~ENDC	81
CREF-80	7, 120
CSEG	18, 42
D	
DB	32
DC	33
DEFB	32
DEFL	39
DEFM	32
DEFS	34
DEFW	35
DS	34
DSEG	18, 43
DW	35

E		INCLUDE	50
END	49	IRP~ENDM	70
ENTRY	38	IRPC~ENDM	72
EQ	27		
EQU	36	L	
EXITM	73	LE	27
EXT	37	LIB-80	7, 128
EXTERN	37	LOCAL	74
EXTERNAL	37	LOW	26
		LT	27
G		M	
GE	27	MACLIB	50
GLOBAL	38	MACRO~ENDM	65
GT	27	MSX•L-80	7, 102
		MSX•M-80	7, 85
H		MSX•M-80 擬似命令	141
HIGH	26	MOD	26
		N	
I		NAME	51
IF	81	NE	27
IF1	81	NOT	27
IF2	81	NUL	25
IFB	82		
IFDEF	81	O	
IFDIF	83	OR	27
IFE	81	ORG	45
IFF	81		
IFIDN	82		
IFNB	82		
IFNDEF	81		
IFT	81		
IF _{xxxx} ~ELSE~ENDIF	81		

P	
PAGE	55
PC モード	40
PUBLIC	38
R	
REPT~ENDM	69
S	
SET	39
SHL	26
SHR	26
SUBTTL	56
T	
TITLE	55
TYPE	25
X	
XOR	28

ア	
アーギュメント	20
アーギュメントフィールド	13
エクスターナルシンボル	16
演算子	25
エントリポイント	6
オブジェクトファイル	6, 87
オブジェクトプログラム	6
オブジェクトモジュール	6
オペコード	19, 145
オペランド	20
オペレーション	19
オペレーションフィールド	13
オペレータ	25
オペレータ定義	25
カ	
外部参照	6
外部シンボル (エクスターナルシンボル)	16
カレントプログラムカウンタ・シンボル	24
擬似命令	19, 29
グローバルリファレンス	6
クロスリファレンサ	7, 120
クロスリファレンス・ファイル	86, 120
クロスリファレンス・リスティングファイル	121
コード相対モード	18, 42
コメントフィールド	13

サ

実行可能ファイル	103
条件擬似命令	80
条件付きアセンブル	10
シンボル	14, 23
シンボル定義	31
シンボルフィールド	13
スイッチ (LIB-80)	136
スイッチ (MSX・L-80)	106
スイッチ (MSX・M-80)	92
数値	20
ステートメント	13
絶対モード	18, 41
ソースファイル	6, 12, 87
ソースプログラム	6, 12

タ

データ相対モード	43
データ定義	31

ハ

パス	86
パブリックシンボル	15
反復	69, 70, 72
反復擬似命令	68
プログラムカウンタ・モード	18

マ

マクロ	19, 64
マクロアセンブラ	7
マクロ演算子	75
マクロ機能	10
マクロ定義	64, 65
命令セットの選択	29
モード	17
文字定数	22
モジュール	6

ラ

ライブラリファイル	103, 129
ライブラリマネージャ	7, 128
ラベル	15
リスティング擬似命令	54
リスティングファイル	87
リロケータビリティ	10
リロケータブル	7
リンクローダ	7, 102
リンクロード	7

お問い合わせについて

弊社では厳重に梱包した上、細心の注意を払って製品を発送しております。万一、輸送上のトラブルが起こった場合にはご一報いただければ新しいものと交換いたします。

マニュアル作成にあたり、なるべく詳細な説明をするように心がけたつもりですが、理解できないところは、実際にコンピュータと向きあって納得のいくまで確かめてください。また、他のページを参照するのもひとつの方法です。それでも疑問点が解決できないときは、購入された販売店に問い合わせるか、(株)アスキー ユーザーサポート(直通電話 03-498-0205)までお電話いただければ、係がお答えします。しかしながら、回線が混み合いご迷惑をかけることもありますので、なるべくお手紙にてお願いいたします。その際には、下記の要領で記入してください。記入されていない項目が1つでもありますと、回答できかねる場合があります。充分注意してください。

また、本製品以外に対してのご意見、ご希望がございましたら、弊社までお寄せください。

<< 記 >>

1. 送付先 〒107 東京都港区南青山6-11-1 スリーエフ南青山ビル
株式会社アスキー ユーザーサポート係
TEL 03-498-0205 (祝祭日を除く月～金曜日、10:00～12:00、13:00～17:00)

2. 必要項目

(1)お客様の氏名、住所(郵便番号)、電話番号(市外局番も含む)

(2)製品名、製品シリアル番号

(3)機器構成

本体装置名、メモリバイト数

CRT装置名、フロッピーディスク装置名

プリンタ装置名

その他I/O、I/F装置名

(4)お問い合わせ内容

お問い合わせの内容は、できるだけ製品のマニュアルに記述されている用語を用いて、具体的にかつ明確に記述してください。なお、障害と思われる現象については、その現象を再現可能な情報が必要です。当社で再現できないものは、調査ができません。その現象が発生するまでの操作手順、データを必ず添付してください。データディスクがある場合は、そのコピーも同封いただくと調査がスピーディになります。

また、お客様固有と思われるアプリケーションの設計、作成、運用、保守については、当社のサポート範囲外ですので、お問い合わせいただいても回答できません。ご了承くださいますようお願いいたします。

**MSX-DOS TOOLS
USER'S MANUAL**

1987年 3月 1日 第1版第1刷発行

監修／編集 株式会社アスキー システム本部

発行所 株式会社アスキー
〒107 東京都港区南青山6-11-1 スリーエフ南青山ビル
振替 東京 4-161144
TEL (03)486-7111

バックアップディスクの作成について

MSX-DOS TOOLSを初めて起動した時には、必ずバックアップディスクを作ってください。ディスクにはいつどんな事故が起こるかわかりません。大切なマスターディスクが破損して読み書きできなくなってしまうは大変です。予備のディスクを作り、マスターディスクは保管しておいて通常はこちらの方を使うようにしてください。予備のディスクは、“format” “copy” の2つのコマンドを使って、簡単に作ることができます。

バックアップディスクの作成は次の手順で行います。

①マスターディスクのライト・プロテクト・ノッチを書き込み不可側に動かします。(ライト・プロテクト・ノッチについては、マニュアル「MSX-DOS-II-3ライト・プロテクト・ノッチについて」を参照してください。)

②マスターディスクをドライブAに入れます。ここで“ドライブA”とは、次のようなドライブを指します。

※接続しているドライブが1台の場合は、そのドライブ。

※接続しているドライブが2台の場合は、ディスク・インターフェイス・カートリッジに接続されているドライブ。

※なお本体内蔵型のドライブを使用している方は、マニュアル「MSX-DOS-II-11 複数のドライブを用いる場合のドライブ番号」を参照してください。

③新しいディスクのライト・プロテクト・ノッチが書き込み可側になっていることを確認してドライブBに入れます。ドライブが1台の場合は、ライト・プロテクト・ノッチの確認だけを行って、ドライブには入れないでください。

④プロンプト (A>■の状態) よりformat [F]と入力します。“format” コマンドについてはマニュアル「MSX-DOS-II-39 FORMAT」を参照してください。

⑤Drive name? (A, B) とフォーマットされる方のドライブ名をたずねてきますので、Bと入力します。

⑥2DDのドライブを使用している場合は、このあと次のようなメッセージが表示されます。これは2DDのタイプの一例なので、もしこれと違うメッセージが表示された時は、そのディスク・ドライブのマニュアルを参照してください。

```
A>format [F]
Drive name? (A, B) B
Drive type
1... single sided, 8 sectors
2... single sided, 9 sectors
3... double sided, 8 sectors
4... double sided, 9 sectors
```

? ■ ←フォーマットの形式を選ぶ (フォーマットの形式についてはマニュアル「MSX-DOS-II-40 ディスクの様式」を参照してください。)

⑦Strike a key when readyと表示されたら、ドライブが2台の場合は、Bドライブに新しいディスクが入っていることを確かめてから何かキーを押してください。ドライブが1台の場合は、マスターディスクをドライブから抜き、これからフォーマットされる新しいディスクを挿入してから、何かキーを押してください。

⑧Format complete と表示されるとフォーマットは終了です。(機種によって違いがありますが、フォーマットにはだいたい1～5分程度かかります。)

⑨“copy” コマンドを起動し、マスターディスクを新しいディスクにコピーします。(TOOLコマンドの“diskcopy”を使ってコピーすることも可能です。その場合はマニュアル「TOOLS-17 DISKCOPY」を参照してください。)

⑩プロンプトよりcopy [F]*.* [B]: [F]と入力します。(ドライブAにあるすべてのファイルをドライブBにコピーするという意味です。)

⑪<ドライブが2台の場合>

```
A>copy *. * b:
MSXDOS      SYS
COMMAND     COM
...
SEARCH      BAS
          45 files copied
```

上記のように表示され、これでコピーは終了です。(機種によって違いがありますが、コピーにはだいたい5～10分程度かかります。)

<ドライブが1台の場合>

```
A>copy *. * b:
MSXDOS      SYS
COMMAND     COM
...
LIB80       COM
MED         COM
Insert diskette for drive B:
and strike a key when ready■
```

上記のようなメッセージが表示されます(この間15秒～1分)。このメッセージが表示されたら、マスターディスクをドライブから抜き、フォーマットされた新しいディスクを、ライト・プロテクト・ノッチが書き込み可側になっていることを確認してからドライブに入れ、何かキーを押してください。

なおディスクを差し換える際には、ドライブのアクセス・ランプが点灯していないことを確認してからディスクを差し換えてください。ドライブがディスクにアクセスしている最中にディスクを抜いてしまうと、ディスクが破損してしまう危険性があります。

新しいディスクへの書き込みが終了すると(この間15秒～1分程度)、次のようなメッセージが表示されます。

```
Insert diskette for drive A:
and strike a key when ready■
```

このメッセージが表示されたら、新しいディスクをドライブから抜き、再度マスターディスクを、ライト・プロテクト・ノッチが書き込み不可側になっていることを確認してからドライブに入れ、何かキーを押してください。この場合も、ドライブのアクセス・ランプが点灯していないかどうか注意して差し換えてください。

この作業を7回ほど繰り返しますと、最後に次のメッセージが出てコピーは終了します。

```
Insert diskette for drive B:
and strike a key when ready
          45 files copied
```

※尚、ドライブが1台の方は、必ずマニュアル「MSX-DOS-II-10 仮想ドライブ機能」をご覧ください。

- ⑫これでバックアップディスクができました。実際にバックアップができたかどうか、“dir” コマンドを使い両方のディスクのディレクトリを見て確認してみてください。(“dir” コマンドについてはマニュアル「MSX-DOS-II-37 DIR」を参照してください。)

- ⑬最後にバックアップディスクを起動させてみて、正しくバックアップが作れたかを確認します。一度本体の電源を切り再び電源を入れるか、またはリセットし、バックアップディスクをドライブAに挿入します。マスターディスクの時と同じように起動すれば、バックアップディスクは完全です。起動しなかった場合は、この手順でもう一度やり直してください。

この度、弊社のソフトウェア製品（同封の磁気媒体やマニュアルなどの印刷物に記録又は記載された情報のことをいい、以下、ソフトウェア製品とさせていただきます。）を御購入いただきましてまことにありがとうございます。

弊社では、お客様に対しまして、下記の内容のソフトウェア製品使用許諾書（以下、本書といたします。）を設けさせて頂いております。

弊社といたしましては、お客様が本書の内容につきご理解を頂き、ソフトウェア製品をご使用頂くことを希望いたしております。

お客様は、ソフトウェア製品をご使用になる前に本書を充分にお読みにになり、その内容につきご理解いただきましたならば、同封の「ユーザー登録カード」を弊社までご返送下さい。

弊社では、本書の内容についてご理解頂き、同封の「ユーザー登録カード」を御返送頂いたお客様につきまして、本書所定のアフターサービスをいたしております。

この「ユーザー登録カード」をご返送頂けない場合には、弊社といたしましては所定のアフターサービスをいたしかねますので、よろしくご了承のほどお願い申し上げます。

【ソフトウェア製品使用許諾書】

株式会社アスキー（以下、アスキーとします。）はお客様に対しまして、以下の内容でソフトウェア製品の非独占性、非譲渡の使用許諾をいたします。

I（使用許諾）

アスキーはソフトウェア製品の著作権又はその製品化に必要な諸権利を保有しております。アスキーは当該権利に基づき、お客様に対して、ソフトウェア製品に関して次の内容の使用権を許諾いたします。

ソフトウェア製品をお客様ご自身が、お客様が保有するコンピュータ上で使用（入力、記憶、転送、出力）する権利。但し、当該使用許諾は、お客様が購入したソフトウェア製品につき、アスキーがそれを稼働させる機器として指定しているコンピュータについてのみとさせていただきます。

II（使用許諾の期間）

当該使用許諾は、お客様がソフトウェア製品の購入した時より発効し、アスキーがお客様に対して、事前の通知を出すことにより終了させるか、または、お客様が本書に記載される事項に違反したことによる使用許諾の終了のときまでとします。

III（禁止事項）

お客様は、次に記載する事項をしてはなりません。

- 1) ソフトウェア製品（マニュアルを含む）の全部又は一部を、媒体の如何を問わず複製すること。また、複製数を超えて複製すること。
- 2) ソフトウェア製品の内容（プログラム）を変更、改作すること。

IV（アフターサービス）

お客様から返送された「ユーザー登録カード」に基づき、アスキーはソフトウェア製品につき、次の内容でアフターサービスいたします。

- ① ソフトウェア製品の媒体（磁気ディスクやテープ等）の不良（磁気によるプログラムの破壊、欠損）のために、ソフトウェア製品が購入時において正常に作動しない場合には代替品と交換します。
- ② ソフトウェア製品の購入から1年以内にアスキーがプログラムの瑕疵（バグ）を修正した改訂版ソフトウェア製品を発表したときには、それに関する情報を提供いたします。
- ③ ソフトウェア製品の機能の追加を行った場合にその旨の通知をします。
- ④ ソフトウェア製品の使用方法等につき電話及び封書における質問を受付ます。

V（第三者の使用）

お客様はソフトウェア製品及びその複製物を販売、頒布、貸与、移転その他の方法で、第三者に使用させることはできません。

VI（アスキーの免責）

アスキーは、お客様がソフトウェア製品を使用して作成した結果（著作物、その他の創作物）について、一切の責任及び義務から免れるものとします。お客様が本書の一事項にでも違反した場合には、アスキーはソフトウェア製品に関する一切の保障及び本書所定の責任及び義務から免れるものとします。

1. 環境の持続可能性を確保する。環境負荷の低減、資源の有効利用、再生可能エネルギーの活用などを通じて、環境への負荷を軽減し、持続可能な社会の実現を目指す。

2. 社会の持続可能性を確保する。社会正義の実現、多様性の尊重、従業員の福祉の向上などを通じて、社会への貢献を最大化し、持続可能な社会の実現を目指す。

3. 経済の持続可能性を確保する。経済成長の促進、イノベーションの推進、競争力の向上などを通じて、経済への貢献を最大化し、持続可能な社会の実現を目指す。

4. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

5. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

6. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

7. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

8. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

9. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

10. 環境、社会、経済の持続可能性を確保する。環境負荷の低減、社会正義の実現、経済成長の促進などを通じて、持続可能な社会の実現を目指す。

【環境負荷低減の取り組み】

当社は、環境負荷の低減に取り組んでいます。具体的には、以下のような取り組みを行っています。

1. エネルギー効率の向上

エネルギー効率の向上に取り組んでいます。具体的には、以下のような取り組みを行っています。

・省エネルギー機器の導入：LED照明、省エネルギーエアコンなど、省エネルギー機器を導入し、エネルギー消費量を削減しています。

・再生可能エネルギーの活用：太陽光発電、風力発電など、再生可能エネルギーを活用し、環境負荷を低減しています。

・エネルギー管理システムの導入：エネルギー管理システムを導入し、エネルギー消費量をリアルタイムで監視・制御しています。

・エネルギー効率の向上：エネルギー効率の向上に取り組んでいます。具体的には、以下のような取り組みを行っています。

・省エネルギー機器の導入：LED照明、省エネルギーエアコンなど、省エネルギー機器を導入し、エネルギー消費量を削減しています。

・再生可能エネルギーの活用：太陽光発電、風力発電など、再生可能エネルギーを活用し、環境負荷を低減しています。

・エネルギー管理システムの導入：エネルギー管理システムを導入し、エネルギー消費量をリアルタイムで監視・制御しています。

2. 廃棄物の削減

廃棄物の削減に取り組んでいます。具体的には、以下のような取り組みを行っています。

・廃棄物の削減：廃棄物の削減に取り組んでいます。具体的には、以下のような取り組みを行っています。

・省資源製品の採用：省資源製品を採用し、資源消費量を削減しています。

・リサイクル品の活用：リサイクル品を活用し、資源消費量を削減しています。

3. 温室効果ガスの削減

温室効果ガスの削減に取り組んでいます。具体的には、以下のような取り組みを行っています。

・温室効果ガスの削減：温室効果ガスの削減に取り組んでいます。具体的には、以下のような取り組みを行っています。

・省エネルギー機器の導入：LED照明、省エネルギーエアコンなど、省エネルギー機器を導入し、エネルギー消費量を削減しています。

・再生可能エネルギーの活用：太陽光発電、風力発電など、再生可能エネルギーを活用し、環境負荷を低減しています。

・温室効果ガスの削減：温室効果ガスの削減に取り組んでいます。具体的には、以下のような取り組みを行っています。

・省エネルギー機器の導入：LED照明、省エネルギーエアコンなど、省エネルギー機器を導入し、エネルギー消費量を削減しています。

・再生可能エネルギーの活用：太陽光発電、風力発電など、再生可能エネルギーを活用し、環境負荷を低減しています。

